# SPEECH UNDERSTANDING SYSTEMS

William A. Woods, et al

Bolt Beranek and Newman, Incorporated

Prepared for:

Office of Naval Research
Advanced Research Projects Agency

November 1975

BBN Report No. 3188
A.I. Report No. 39



# SPEECH UNDERSTANDING SYSTEMS

Annual Technical Progress Report
30 October 1974 to 29 October 1975

Quarterly Technical Progress Report No. 4
1 August 1975 to 29 October 1975



Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904



This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: BBN Report No. 3188 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE (and Subtitle): SPEECH UNDERSTANDING SYSTEMS<br>Annual Technical Progress Report<br>30 October 1974 to 29 October 1975 | 5. TYPE OF REPORT & PERIOD COVERED: Annual Tech. Prog. Report<br>30 Oct. 1974 to 29 Oct '75 |
| | 6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 3188 |
| 7. AUTHOR(s): W. Woods, M. Bates, G. Brown, B. Bruce, C. Cook, L. Gould, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue | 8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0533 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc.<br>50 Moulton Street<br>Cambridge, MA 02138 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 5D30 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: ONR<br>Department of the Navy<br>Arlington, VA 22217 | 12. REPORT DATE: November 1975<br>13. NUMBER OF PAGES: 115 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report): Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acoustic-phonetic recognition, audio-response generation, beta distributions, data base, dictionary expansion, discourse model, fact retrieval, formal command language, formant tracking, labeling, lexical retrieval, likelihood ratio, multi-component systems, natural language understanding, natural language retrieval system, parametric modeling, parsing, phonology,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes recent progress of the BBN speech understanding system project covering the period from October 1974 to October 1975. The BBN speech understanding project is an effort to develop a continuous speech understanding system which uses syntactic, semantic and pragmatic support from higher level linguistic knowledge sources to compensate for the inherent acoustic indeterminacies in continuous spoken utterances. These knowledge sources are integrated with sophisticated signal processing (cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. Key Words - cont'd.

phonological rules, phonological rule tester, pragmatics, prosodics, question-answering, recognition strategies, resource allocation, segmentation, semantic interpretation, semantic networks, signal processing, speech, speech recognition, speech understanding, synthesis-by-rule, system organization, vocal tract length, word verification.

---

20. Abstract - cont'd.

and acoustic-phonetic analysis of the input signal, to produce a total system for understanding continuous speech. The system contains components for signal analysis, acoustic parameter extraction, acoustic-phonetic analysis of the signal, phonological expansion of the lexicon, lexical matching and retrieval, syntactic analysis and prediction, semantic analysis and prediction, pragmatic evaluation and prediction, and inferential fact retrieval and question answering, as well as synthesized text or spoken output. The report summarizes the fourth year of a five-year development effort.



1a/

SPEECH UNDERSTANDING SYSTEMS

Annual Technical Report
30 October 1974 to 29 October 1975

Quarterly Technical Progress Report No. 4
1 August 1975 to 29 October 1975

ARPA Order No. 2904

Program Code No. 5D30

Name of Contractor:
Bolt Beranek and Newman Inc.

Effective Date of Contract:
30 October 1974

Contract Expiration Date:
29 October 1975

Amount of Contract: $1,041,261

Contract No. N00014-75-C-0533

Principal Investigator:
William A. Woods
(617) 491-1850 x361

Scientific Officer:
Marvin Denicoff

Title:
SPEECH UNDERSTANDING SYSTEMS

QTPR Editor:
Bonnie Nash-Webber
(617) 491-1850 x227

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

Table of Contents

page

I. PROGRESS OVERVIEWS

A. System Structure . . . . . 1
B. Signal Acquisition and Analysis . . . . . 7
C. Acoustic-Phonetic Recognition . . . . . 11
D. Dictionary . . . . . 14
E. Phonology and Dictionary Expansion . . . . . 17
F. Lexical Retrieval . . . . . 19
G. Syntax . . . . . 21
H. Semantics . . . . . 28
I. User and Discourse Model . . . . . 31
J. Audio-response Generation . . . . . 33
K. The Travel Budget Manager's Assistant . . . . . 34
L. Prosodics . . . . . 37
M. Verification . . . . . 38
N. Recognition Strategies . . . . . 41
O. PDP11/SPS41 Signal Processing System . . . . . 46

References . . . . . 48

II. TECHNICAL NOTES

A. Parametric Modeling of Probability Distributions . . 50
B. Digital Spectrograms . . . . . 66
C. A Phonological Rule System for Dictionary Expansion. 68
D. Audio-response Generation . . . . . 95

# I. PROGRESS OVERVIEWS

## A. System Structure

In the past year, the Travel Budget Manager's Assistant under development by the BBN Speech Understanding Project has evolved into a dual-purpose speech and text understanding system. Conceptually, the system is organized as shown in Figure 1, with the former "Control" component of SPEECHLIS now serving the role of speech understanding controller, directing the Syntax, Semantic and Lexical Retrieval components towards forming the best possible model of an input utterance. Under this organization, the Travel Budget Manager's Assistant functions as the System Controller. It initiates a dialogue with the system user, who may respond at any point with either spoken or typed input.

The four components on the left-hand side of Figure 1 (Syntax, Semantic Interpreter, Retrieval and Audio-Response Generator), together with the System Controller, function as a complete text understanding system. If the manager responds to his assistant in text, his response is sent directly to the Syntactic component to be parsed. The resulting parse tree, or trees, are then sent to the Semantic Interpreter, who tries to find one or more consistent interpretations for them in our formal retrieval

language. (If there turns out to be more than one consistent interpretation at this level, the manager is currently asked to resolve the ambiguity.) The formal interpretation is then sent to the Retrieval component for execution and response. If the system decides to respond orally, the Audio-Response component is activated; otherwise the response is typed. Often it may happen that the Retrieval component needs to ask a question of the user before completing the execution of the current utterance. If so, the execution is suspended, the question asked, and control returned to the System Controller to await the user's reply, either spoken or typed. If the reply is satisfactory, the suspended execution is then resumed.

When the user talks, rather than types, to the system, its entire resources, including those used in text understanding, are called upon to acquire, process, understand and act on his request. In processing the spoken input, the System Controller first activates the real-time interface RTIME to acquire the incoming speech signal, then the signal processing component DPSA to compute the relevant parameters of the signal, and then the APR component to produce a segment lattice representation of the utterance for input to the Lexical Retrieval component. Control is then turned over to the Speech Understanding Controller, the former "Control" component, which attempts to use the Syntactic, Semantic, Lexical Retrieval and Verification

components to arrive at the best model of the utterance that is both consistent with the acoustics and syntactically and semantically valid. The result of this processing is either nothing, meaning no such model could be found, or the parse tree(s) produced by Syntax. The system thereafter functions as for typed input, interpreting, executing and responding to the Manager's request.

The conceptual structure of the SPEECHLIS system is being implemented as a set of nine interacting processes, or forks, on TENEX, in the configuration shown in Figure 2. Each box corresponds to a single TENEX fork, with the lines representing the communication links between them. Where several components reside in the same fork, these links are not explicitly shown.

The reasons for this particular mapping of conceptual structure into fork structure derive from:

1. differences in the implementation languages of the various components (e.g., The Lexical Retrieval component is implemented in BCPL, the Signal Processing component in Fortran, the higher level components in Interlisp, etc.);

2. storage demands (There is not enough room in one TENEX fork for all the higher level components.);

3. interaction demands (i.e., Where components interact frequently, we have tried to have them reside in the same fork to cut down the costs of fork switching and information passing);

4. historical circumstance. (This is not as frivolous as it might appear. Work on signal acquisition, signal processing, acoustic-phonetic recognition, acoustic verification, lexical matching, and audio response began, and has continued, as individual stand-alone

programs, with which their creators interact in the course of development and testing. A small amount of additional control logic for system linkage allows these programs to function both as parts of the complete speech understanding system or as stand-alone developmental tools).

The circular objects in Figure 2 represent open files which are accessed by several components during run-time. Such files save address space and eliminate the need to have multiple copies of information and avoid the possible inconsistencies that could arise from having them. The semantic network files (see I.H.) are currently accessed by the Semantic Recognition and the Retrieval components. (In the coming year, we plan to enable the Semantic Interpreter to use the network as well.) The Dictionary is accessed by all the higher level components.

In the past year, we have implemented all the system components, all the communication links between those components residing together, and also those links shown as solid lines in the figure. Those shown as broken lines have so far only been simulated, but we plan to implement them very early in the coming year. With this much of the system active, we have been able to run and experiment with the entire text-based system, as well as to develop strategies for using the Syntax, Semantics, and Lexical Retrieval components to understand spoken utterances. This is discussed more fully in Section I.N.

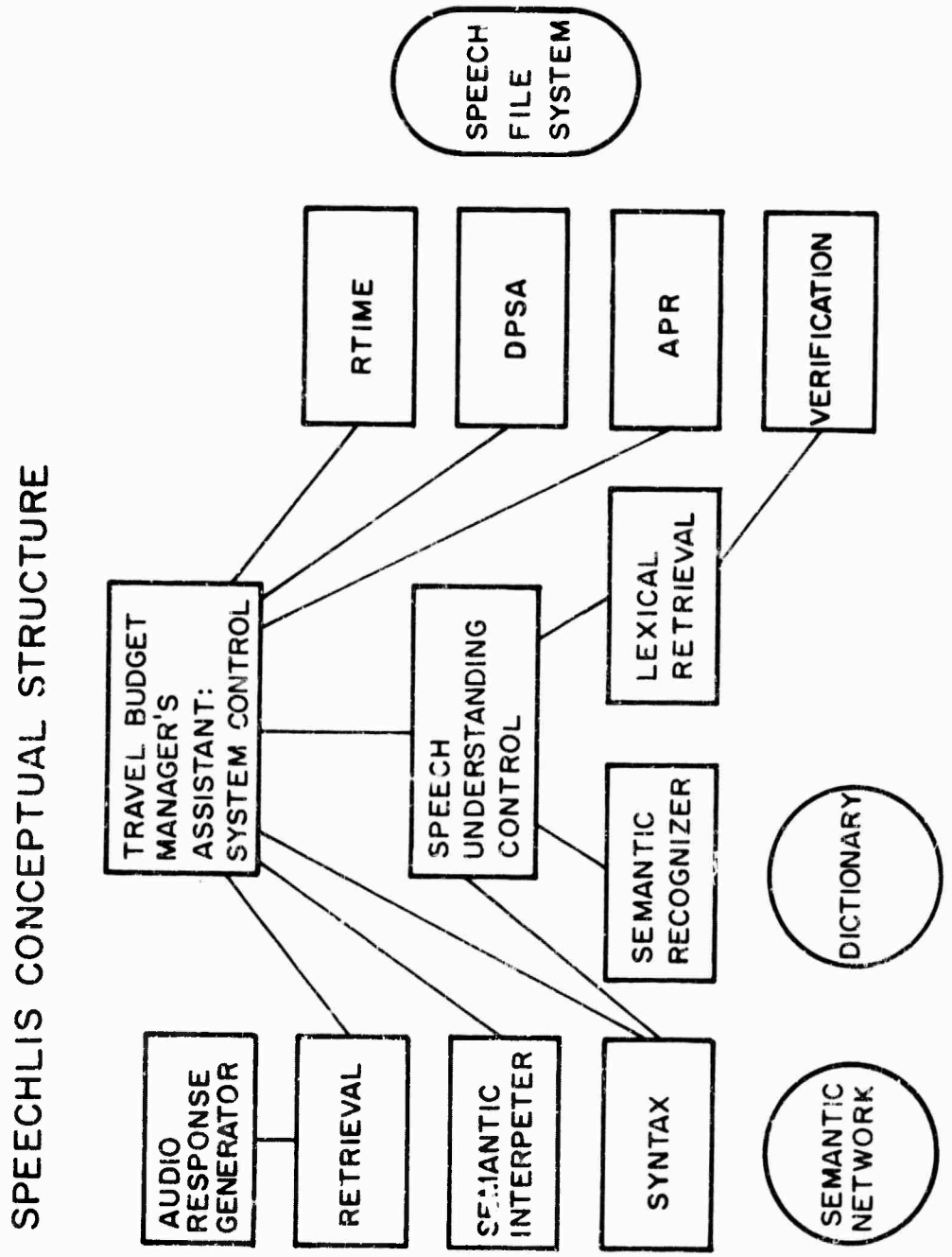
SPEECHLIS CONCEPTUAL STRUCTURE
SPEECH FILE SYSTEM
RTIME
DPSA
APR
VERIFICATION
TRAVEL BUDGET MANAGER'S ASSISTANT: SYSTEM CONTROL
SPEECH UNDERSTANDING CONTROL
LEXICAL RETRIEVAL
SEMANTIC RECOGNIZER
DICTIONARY
AUDIO RESPONSE GENERATOR
RETRIEVAL
SEMANTIC INTERPETER
SYNTAX
SEMANTIC NETWORK

Figure 1

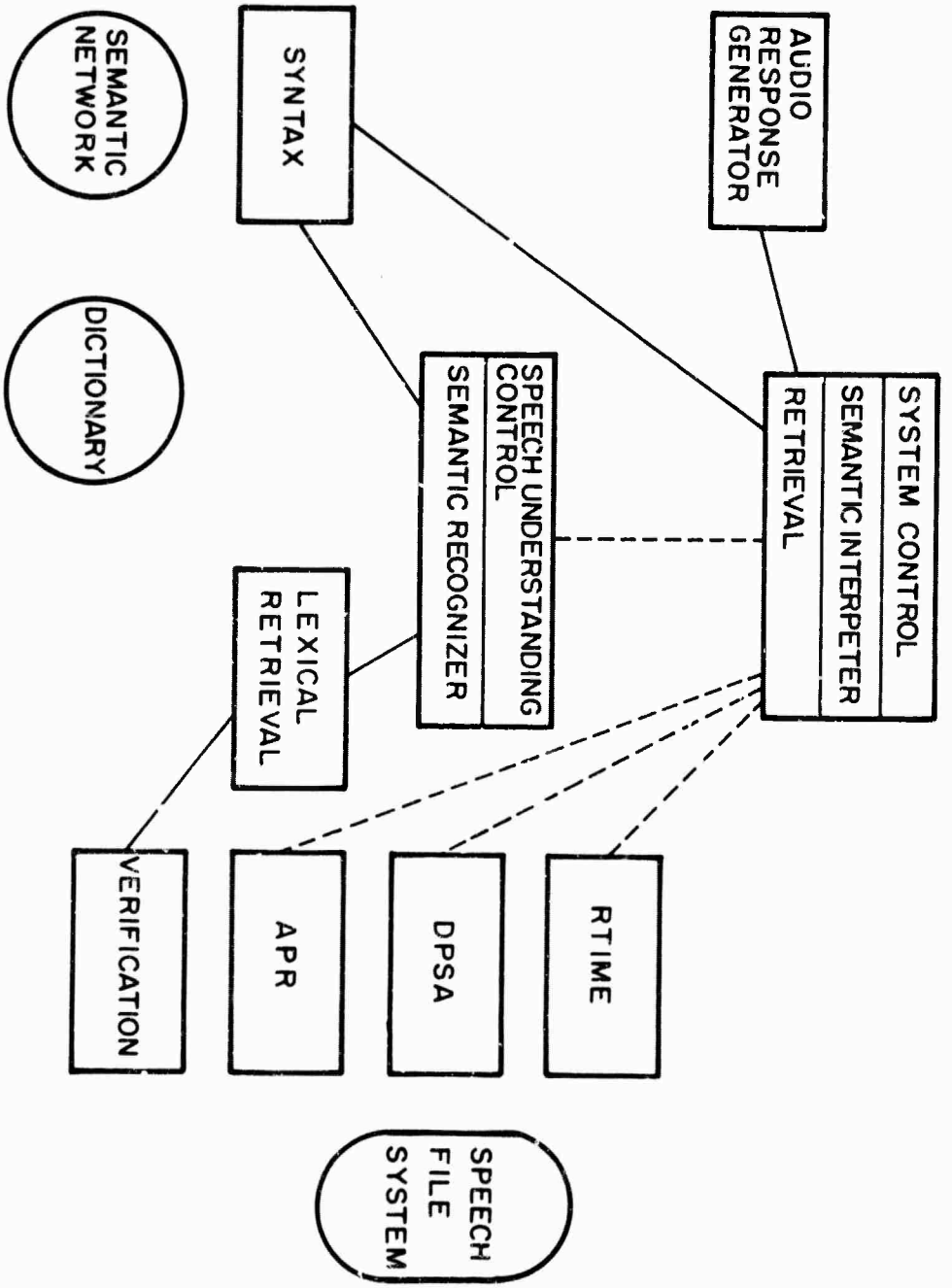
SPEECHLIS FORK STRUCTURE
SYSTEM CONTROL
SEMANTIC INTERPETER
RETRIEVAL
AUDIO RESPONSE GENERATOR
SYNTAX
SEMANTIC NETWORK
SPEECH UNDERSTANDING CONTROL
SEMANTIC RECOGNIZER
DICTIONARY
LEXICAL RETRIEVAL
VERIFICATION
APR
DPSA
RTIME
SPEECH FILE SYSTEM

Figure 2

## B. Signal Acquisition and Analysis

### 1. RTIME

Work on RTIME, the speech waveform sampling/editing/playback program, has progressed in two directions. We have improved its ability to detect the beginning and endpoint of an utterance by replacing its utterance beginning/endpoint location algorithm with a new one, based on summed amplitude and zero-crossing rate, essentially as described in [Rabiner and Sambur, 1975]. In addition, we have integrated RTIME, which is primarily an interactive program, into the SPEECHLIS system via the addition of a "SPEECHLIS" command. This allows RTIME to be invoked as an automatic utterance-acquisition module within the context of the complete speech understanding system.

### 2. Signal Processing

In the past year, two new parameter calculation methods have evolved to meet specific needs of the Acoustic-Phonetic Recognition and Acoustic Verification modules. A very simple technique has been discovered which suffices for estimating formants to very good accuracy. For each analysis frame (20 msec wide, at 10 msec intervals), the three poles of the digitally preemphasized signal having the lowest bandwidths (subject to only a few frequency constraints) are designated as the formants. No previous segmentation of the utterance or frame-to-frame constraints

are required. We have found that this technique, in combination with a 3-point median smoother, yields excellent results. In labeling fricatives and unvoiced plosive bursts, we have found that a useful correlate of place of articulation is the lower extent of the high frequency frication energy. We have therefore defined a class of "Center of Mass" or "CM" parameters. The parameter CM50 would be that frequency above which lies 50% of the energy in the 0-5 kHz preemphasized spectrum (its center of mass). For numbers other than 50%, this is, of course, not the true center of mass, but a weighted one. We have adopted CM75 for characterizing these segments.

The signal processing component of SPEECHLIS, named DPSA, has also undergone consolidation and integration. The new parameter computation algorithms described above were incorporated into DPSA so that it would compute all parameters required for the Acoustic-Phonetic Recognition module. As with RTIME, a "SPEECHLIS" command was also added to DPSA to allow it to be invoked automatically by the control component of SPEECHLIS and thus function as an integral part of the system.

## 3. F0 Extraction

We noted in [Woods et al., 1975b, pp. 26-27] that our current fundamental frequency extraction routine made too many errors for its results to be used by the BOUND3

prosodic boundary detection program supplied by the UNIVAC group. Consequently, work on a new F0 extractor was undertaken, employing an algorithm essentially identical to the downsampled and center-clipped autocorrelation algorithm described by Dick Gillmann of SDC [Gillmann, 1975]. The results observed so far are much better than those of the previous routine, particularly in the regions of unvoiced/voiced and voiced/unvoiced transitions. A side benefit is the fact that the computation time required is reduced by about a factor of 4, due to the downsampling. We expect to be able to incorporate the new F0 extractor into DPSA in the near future, which will greatly facilitate the work on prosodic boundary detection.

## 4. Analysis Methods

We have been investigating two other analysis techniques, based on linear prediction, for possible application in the acoustic analysis phase of SPEECHLIS, one dealing with speaker normalization, and one dealing with linear prediction analysis itself.

In order to enrich our repertoire of tools for speaker normalization, we have implemented an algorithm that estimates the instantaneous vocal tract length of the speaker for each vowel frame. The algorithm is based on work by Wakita [1975] and Zue and Paige [1975]. Briefly, the procedure starts from the frequencies and bandwidths of

the first three formants. From these and an assumed sampling frequency, a vocal tract log area function is computed. This process is iterated to find the sampling frequency that minimizes the variance of the log area function. From this optimum value, an effective vocal tract length is easily computed. Further experimentation is required to determine if this technique will be more effective in speaker normalization than our present technique, which uses the average fundamental frequency of the utterance [Schwartz, 1971].

One of the main problems in the accurate estimation of formants and signal energy is the variability in the pitch of an individual speaker as well as its variability across speakers. The autocorrelation method of linear prediction, which we have been using so far, has the disadvantage that it is sensitive to wide variations in pitch, due to the interaction between the analysis window and the pitch period. The covariance method does not use a window and hence does not exhibit the same degree of sensitivity to pitch variations. However, it has the disadvantage that the stability of the computed model is not assured. We are now working on a class of lattice methods which do not require windowing and yet do preserve stability. In addition, these methods would be computationally efficient, unlike the lattice method of Itakura [1971].

## C. Acoustic-Phonetic Recognition

Based on experience gained during the past year, we have completely rebuilt the APR component, utilizing the theoretical procedures described in the BBN Final Report [Woods et al., 1974].

The new APR component starts by looking at dips in three energy parameters to form a preliminary segment lattice. Then, a sequence of 22 ordered Acoustic-Phonetic rules is applied to the lattice. The rules only apply at certain places in the lattice, depending on the partial segmentation and labeling. Using this labeling in the lattice and the acoustic parameters, the rules can delete branches, expand the lattice by adding branches, and change or narrow the label on any segment. It is of some interest that the ordering of the rules, so far, has been relatively straightforward.

The program currently distinguishes all vowels and diphthongs, unvoiced plosives, intervocalic glides, strident and weak fricatives, affricates, and flapped dentals. It also finds and labels prevocalic and postvocalic glides using formant transitions. Formant transitions are also used to separate postvocalic velar consonants from non-velars. The program also detects unreleased plosive-plosive pairs, pauses, syllabic nasals, and [stressed tense vowel] - [unstressed lax vowel] pairs, such

as IY-AX in "give me a list." The program associates with each segment a number which is equal to the maximum energy in the segment. This is used by the word matcher to evaluate within-word stress differences. The APR component currently requires approximately two times real time to generate a segment lattice from the acoustic parameters for an utterance, including the time necessary to read in all the parameters. It is expected that this will remain essentially constant.

The parameters currently used by the program are -

| Name | Definition | Use of Parameter |
|---|---|---|
| LEZ | Smoothed energy in the region from 120-440 Hz. | Sonorant obstruent segmentation, Aid in voicing decision on fricatives |
| MEPZ | Smoothed energy in the preemphasized spectrum from 640-2800 Hz. | Segmentation of non-vowels within sonorant regions. |
| HEPZ | Smoothed energy in the preemphasized spectrum from 3400-5000 Hz. | Segmentation of non-vowels within sonorant regions. |
| ROP | Energy in the preemphasized spectrum. | Burst location, Plosive and Fricative identification, and many others. |
| F1<br>F2<br>F3 | Formant Frequencies | Detecting glides and nasals within sonorant regions, segmenting vowel regions, labeling vowels and glides, some consonants. |
| F0 | Fundamental Frequency | Normalizing formants, aid in voicing decision. |
| CM75 | "75% center of mass" indicates rough spectral shape. | Used in identifying fricatives and unvoiced plosives. |

In developing the algorithms used in the APR, we have made extensive use of our Acoustic-Phonetic Experiment Facility, APEF [Woods et al., 1974, pp. 53-55; Woods et al., 1975a, pp. 20-32] on a data base of utterances within our travel budget management domain. A total of 70 utterances, gathered from 38 sentences spoken by six speakers (4 male, 2 female) have been digitized and stored on-line. Each utterance has been carefully hand-labeled, with time markers synchronized with the waveform and computed acoustic parameters.

The APEF is also used to produce a confusion matrix between segment labels and dictionary symbols, which is used at run-time in transforming a preliminary segment lattice into the form required by the lexical retrieval component. To produce such a matrix, the APEF uses a set of preliminary segment lattice files generated by the APR from the entire data base. Then for a particular preliminary segment lattice file, the label for each segment determines a column of the matrix which contains log likelihood ratios for each of the dictionary symbols.

Development of the APEF continues as we find the need to perform more complex experiments. Some of the more useful improvements in the APEF this past year have been: the ability to display rotatable 3-D scatter diagrams, the ability to specify multiple optional segments in the

phonetic context for an experiment, the inclusion of several techniques for smoothing and modeling probability density functions, and general improvements in flexibility and speed. With respect to the probability density functions, the resulting smoothed parametric models of the relations between acoustic parameter(s) and allophones will be used to selectively modify the likelihood ratio for several of the allophones in a segment label, in order to characterize each segment according to the actual values of the acoustic parameters found.

D. Dictionary

During the past year, the size of the dictionary for the travel budget management domain has increased from 350 base forms to the thousand (actually 1096) required by the SUS Study Group Final Report [Newell et al., 1973]. (A table showing the number of entries for each major syntactic/semantic category is given in Figure 3. Since some base forms belong to more than one category, the sum of the sub-totals is more than 1096.)

Our major considerations in growing the dictionary to this size were to provide the user with a habitable and reasonably complete way of discussing the domain and to provide our acoustic-phonetic front end with a wide range of phonetic and phonological situations. Thus, several methods

were used in selecting new words for the dictionary.

(1) A set of sentences related to the domain was collected, and all words in these sentences and in the sentences from incremental simulations were included in the dictionary;

(2) Roget's Thesaurus was used to find useful synonyms of existing words, to increase the number of ways things could be said;

(3) The first and last names of all BBN Division 4 personnel were added;

(4) As were the names of all Division 4 projects and their sponsors;

(5) Many place names (cities, states, countries, universities) were added. All places appearing as the destinations of trips recorded in the data base are represented in the dictionary; further place names were obtained from meetings announced in the ACM Calendar for the last year.

Not all the new base forms in the dictionary are actually new "words." About 60 of them represent alternative pronunciations of existing base forms, where the occurrence of one form or the other has syntactic import. Fourteen of these are alternate forms for the integer and ordinal "teens" (13, 13th, 14, 14th, etc.). The two separate base forms provide a way of indicating the difference of stress in the words, which depends on their use as a modifier or the head of a syntactic construction. For example, one says "He's thirteen" (head), but "I'm having thirteen people for dinner" (modifier). The first has the base form THIRTEEN-H with stress on the second syllable, while the second has the base form THIRTEEN-M, which has stress on the first

syllable; each is given the appropriate stress in its phonemic representation. Similar pairs of forms are provided for the ordinals.

The rest of the 60 special base forms are "reduced" forms of many monosyllabic articles, auxiliaries, conjunctions, prepositions, and pronouns. These are provided for words whose pronunciations change considerably depending on whether they are stressed or not. For example, the article "a" is represented by two forms:

A pronounced with the vowel EY and stressed

A-R pronounced with the vowel AH and unstressed

There are several kinds of syntactic distinctions which the occurrence of a reduced or non-reduced form may signify. For example, IN may be either a preposition or a verb particle, but IN-R can only be a preposition, as verb particles are always stressed. Similar distinctions may be drawn with regard to some words which can be either verbs or auxiliaries: the reduced form can only be an auxiliary, as main verbs are always stressed. These distinctions are useful to the grammar for disambiguation.

With respect to the inflected forms of words, only irregular ones have been included in the basic thousand-word dictionary. Inflectional rules for forming regular inflections are applied during the dictionary expansion phase (see [illegible]), which currently increases the dictionary

size to 1449 base forms.

The pronunciations in the dictionary have all been changed to conform with those given in Kenyon & Knott, A Pronouncing Dictionary of American English. The symbols used are the 48 phonemes of the modified ARPABET, plus 4 stress indicators: primary (!2), secondary (!1), unstressed (!0) and reduced (!-).

| | | | |
|---|---|---|---|
| ADJ | 74 | QDET | 6 |
| ADV | 46 | QUANT | 14 |
| ART | 13 | QWORD | 5 |
| AUX | 23 | V | 172 |
| CONJ | 14 | SPONSOR | 21 |
| MODAL | 15 | PROJECT | 72 |
| N | 162 | LASTNAME | 108 |
| NEG | 1 | FIRSTNAME | 89 |
| NPR | 65 | CITY | 113 |
| PARTICLE | 7 | STATE | 51 |
| POSS | 14 | COUNTRY | 21 |
| PRECONJ | 2 | MONTH | 12 |
| PREP | 48 | WEEKDAY | 7 |
| PRO | 37 | INTEGER | 36 |
| QADV | 3 | ORD | 30 |

Fig. 3. Distribution of Syntactic Categories.

## E. Phonology & Dictionary Expansion

Over the past year or so, our approach to dealing with phonological variations in continuous speech has undergone considerable change. In our present approach:

1) Phonological rules are written and applied generatively to the lexicon, rather than analytically to the segment lattice;

2) Within-word phonological effects are dealt with by assuming, for each lexical entry, a minimal and appropriate set of baseforms, which are then expanded via a collection of phonological rules into all possible pronunciations;

3) Across-word phonological effects are described by a separate, but similar, set of rules which are applied to the complete set of pronunciations that result from the consideration of within-word phonological variations;

4) Each pronunciation in the lexicon has associated with it a score derived from the relative goodness of its baseform and the likelihood of each applied rule. This is intended to quantify the goodness and likelihood of each pronunciation variant.

The generative phonological rules in the system account for both general phonological effects and ones which are dependent on the performance quality of the APR. This is because the APR itself does not currently compensate for all co-articulatory effects and thus tends to mislabel certain segments. For example, lateralization of the vowel [IH] in the word "list" lowers its second formant by approximately 500 Hz, resulting in the consistently mislabeling of this vowel as [UH]. Since the APR currently does not incorporate such co-articulatory effects, a set of acoustic phonetic adjustment rules was written to further modify the lexicon. Performance of the lexical retrieval component will provide feedback as to the goodness of these front-end-dependent rules.

Minor refinements have recently been made to the baseforms and phonological rules to the extent that the

expanded dictionary has reached a point of stability. The weights associated with the baseforms and rules, on the other hand, are still being modified in order to arrive at an appropriate quantitative measure of goodness for each pronunciation variant.

## F. Lexical Retrieval

The present Lexical Retrieval component, implemented during the last year, differs radically from our previous Lexical Retrieval component [Woods et al., 1974, pp. 56-83] in several areas. First, it was designed as a means of implementing a formal scoring philosophy in which word scores are defined as the probability that they were spoken, given the acoustic evidence. To effect this philosophy at the word level, it was also necessary to implement it at the segment labeling level. Thus, each segment in the segment lattice is labeled with a vector whose entries represent the probability that a particular phoneme was spoken, given the acoustic evidence. Secondly, the present Lexical Retrieval component is designed to operate from a tree structured phonetic dictionary in order to facilitate an efficient handling of explicitly stated word boundary effects. Another benefit of this structure is that considerable matching effort is avoided by runtime pruning on whole sub-trees when such pruning does not affect the final results. The capabilities of the new Lexical Retrieval

component can be summarized as follows:

1) It permits great generality of acoustic input description, in that it accepts an arbitrarily connected segment lattice where each segment is a vector of probabilities of phonemes.

2) It can do word scans in either direction (left-to-right or right-to-left) and can prune (while matching) on the basis of either the immediate or potential word score.

3) It can constrain the word(s) being sought to belong to any group of words (directly specified) or to possess any of a group of properties (typically these are syntax classes) or to have one of a group of specified lengths.

4) It can use all, none, or any combination of these constraints to specify the words being sought.

All of these constraints are checked during the matching operation itself rather than just throwing away words afterwards. This has the advantage of substantially reducing the matching effort. Other indirect constraints can be specified to the Lexical Retrieval component as well, such as the position in the lattice, the context of adjacent words, minimum acceptable score, and minimum rank among the words to be returned.

Work has recently been done to extend the performance of the Lexical Retrieval component with respect to the size of the dictionary it can handle and the factors taken into account in scoring wordmatches. Observed deficiencies have also been eliminated.

With respect to wordmatch scoring, extensions were made to the Lexical Retrieval component so that information associated with a sequence of segments (as opposed to a single one) could be investigated as to its potential usefulness. So far, we have begun to investigate scoring augments due to the relative stress patterns in multi-syllabic words.

With respect to deficiencies, it was observed that the Lexical Retrieval component's handling of word boundary effects was causing problems by only matching the "kernel" of the word and not its ends due to the possible application of specific word boundary rules. Certain wrong words often scored better than the labeling indicated and good words worse. Extending the effective word scoring beyond the kernel to some user indicated position in the transformation part of the word boundary rule required modifications to both the dictionary compiler and the matching algorithm. Although these have not yet been completed, we expect to finish this work early in the next quarter and incorporate it into the system.

G. Syntax

This year has seen advances in both parts of the SPEECHLIS Syntactic component - the parser and the grammar - as well as its interface with the rest of the system. We

now have two grammars and two parsers under parallel development, all within the paradigm of an augmented transition network.

With respect to a grammar, there are now two grammars available to the system. The first, SPEECHGRAMMAR, is the general modified ATN grammar we have been building since the beginning of this project. The second, SMALLGRAM, is a "pragmatic grammar" which uses syntactic/semantic categories to characterize the meaningful utterances within the domain. Both grammars now enable the syntactic component to parse a large variety of utterances not covered before this year, such as date and number expressions, people and place names, and verb-particle sentences.

In general, SMALLGRAM provides much more immediate support to the speech understanding process since it constrains the possible acceptable sentences much more strongly than does SPEECHGRAMMAR. While its task-specific nature means that it would have to be rewritten for a different domain, it has significant advantages for speech understanding. Specifically, it permits a uniform framework for integrating detailed predictions and constraints taking account of information of semantic and pragmatic nature in addition to what would ordinarily be considered as syntactic information. This type of grammar reflects our best judgment of the approach to automatic speech understanding

which will first become available for applications, and we are implementing it in addition to our more general SPEECHGRAMMAR in order to have a prototype of the kind of system which we think will have the first chance of successful application.

SMALLGRAM actually has more states and arcs than the general purpose SPEECHGRAMMAR, due to its splitting up into specific cases classes of constructions which are treated as a unit in the general purpose grammar. This splitting enables the grammar to make different predictions about the possible realizations of subjects of sentences versus objects of verb phrases or prepositional phrases, different predictions about possible verbs as a function of the subject of a sentence, different predictions about objects as a function of the main verb, and even different predictions about possible determiners as a function of position in the sentence, mood of the sentence, and main verb. In addition, the grammar can contain specific tests against global discourse variables to enable specific predictions of possible acceptable utterances as a function of preceding discourse context. Hence, it is effectively a grammar of pragmatically acceptable utterances, characterizing in a single ATN formalism all of the syntactic, semantic, and pragmatic requirements for an acceptable utterance.

SMALLGRAM also embodies some extensions to the grammar formalism which have been made to allow explicit indication of scoping of register setting and testing operations, and a more systematic treatment of feature setting and testing. The scope indications will allow the use of context sensitive information contained in registers at an earlier stage of the recognition process, thus restricting the number false paths followed by the parser by terminating them sooner. These extensions to the grammar formalism have been made in order to support a new syntactic parser which is now under construction, and which we expect will have significant speed advantages over the current syntactic component.

In the past year, we have also worked on developing suitable syntactic structures for input to the semantics and pragmatics components. For example, passive sentences are parsed with their surface structure intact, since only semantics can determine the true subject of sentences like "The money was spent by June" and "The money was spent by John." Similarly, the time and date structures produced by syntax reflect the referent of the expression rather than its syntactic form. Thus "July ten" will produce the same structure as "the tenth of July" [Woods et al., 1975c, pp. 15-26].

With respect to our original speech parser, SPARSER,

our main advance this year has been to include a scoring mechanism; so that as a parse path is constructed, it is scored at each step. Thus alternative paths may be compared relative to one another and "better" (i.e., higher scoring) paths continued before the others.

To implement this mechanism, a change was made to the form of the grammar to include a weight on each arc. This weight can be taken as a rough measure of how likely that arc is to be the correct one to take out of the state to which it is attached. The parser was then modified to employ the weights in the following way. Each configuration created receives a score which is determined by the score on the configuration preceding it and the weight on the arc causing the transition between them. In the simplest case, the score of a new configuration is the sum of the score of previous configurations and the weight on the arc between them. Thus the score on a configuration may be considered the score of the parse path terminating on that configuration. If the arc is a PUSH arc, the score of the constituent used on that arc (which may be semantically as well as syntactically based) is also used to compute the path score.

The parser considers a set of the highest-weighted active configurations at each step and tries to extend them before selecting a new set. Thus good-scoring paths are

extended, while lower-scoring ones remain active but unextended unless better paths fail. This allows some parallelism in the parser, and the dangers of depth-first processing are, to some extent, avoided.

Other changes were made to the parser to reflect our new treatment of inflected words. Until quite recently, a word match for an inflected word was given as a match for the root form, plus a code indicating the inflection. Inflected words are now included in the expanded dictionary, so are matched as themselves. This allows the parser to treat regular and irregular words uniformly, for example, with respect to where it can find their syntactic features.

Recently, the parser has received extensive exercising under the new syntax-driven control strategy. This has both shown up several deficiencies with the parser's prediction mechanism and suggested other modes in which syntax may be run. We expect to correct these deficiencies soon and explore these other modes during the coming months.

With respect to syntax and prosodics, we have had discussions with the UNIVAC group as to how each of our grammars might best use the kinds of prosodic information produced by programs under development there to select or evaluate parse paths. We are also working on plans to use advice from the pragmatics component as to the likely nature of the next utterance to guide SMALLGRAM to a correct

interpretation of the utterance.

Along with our efforts to develop SPARSER, work in syntax has proceeded in the direction of improving parser performance by using a more efficient internal grammar representation. The one we have come up with is produced by a grammar compiler from an ATN grammar, and is so structured that most parsing operations can be done quickly and many others are made unnecessary. We are currently at work on a bi-directional breadth-first parser that will run compiled in BCPL and should produce dramatic increases in speed and savings in space.

With respect to interfacing the Syntactic component with the rest of the system, a long-standing deficiency in Syntax's ability to communicate with the rest of the speech understanding system was corrected this year. The problem lay in Syntax's inability to handle "fuzzy" wordmatches, the mechanism developed to accommodate for word boundary uncertainties at the lexical retrieval level. A "fuzzy" wordmatch is a collection of similar wordmatches which differ slightly with respect to left and right boundaries, but are essentially located in the "same" place in the utterance. The mechanism reduces the combinatorics that would result from treating each member of the fuzzy as a separate, independent wordmatch. However, many of the data structures and procedures used by the Syntactic component

rely on the assumption that wordmatches have a single known left and right boundary, and it was decided that it would be very difficult for Syntax, if that assumption became invalid.

The solution to the problem was to disguise fuzzy wordmatches for Syntax by creating a mock wordmatch to represent the fuzzy one. To Syntax, there is no difference between a mock match and a real one. However when Syntax wants to learn, for example, what wordmatches abut a mock match to its left or right, the answer it gets contains all matches which abut one or more members of the fuzzy wordmatch being represented. These matches may in turn be real or mock ones. The point is that Syntax always sees a string of contiguous wordmatches when it is possible for the rest of the system to see a continuous path through the simple and fuzzy matches contained in a theory.

## H. Semantics

During the past year, our work in semantics has been along five lines: (1) improving the reliability and efficiency of our basic semantic network package (SEMNET); (2) improving the efficiency of our routines for searching the travel network, TRAVELNET, for semantic associations among words and concepts; (3) extending somewhat the range of "understandable" semantic associations; (4) expanding the

number of words and concepts known to TRAVELNET; and (5) building a semantic interpretor to transform the output of the parser into a request in our formal command language (See I.K.).

With respect to the SEMNET package, we replaced our earlier procedure for merging together sons of the same parent network with a procedure that prevents multiple users from making simultaneous changes to the same network. This eliminated the difficult problem of deciding how to recognize and deal with node and link deletions and other incommensurate changes to the network. The new procedure is more reliable, in that a user is guaranteed that the changes he makes to the network will be visible to subsequent users. SEMNET has also been made more efficient by keeping the content of a network on external files with only an index file in-core. This reduces drastically the in-core space requirements of a user's semantic network and the time needed to file new versions [Woods et al., 1975c, pp. 9-11]. Recently, we have expanded our reliance on external storage by also keeping the "terms" (node names) and corresponding pointers on an external file. A term and its corresponding pointer are then only loaded when referenced for the first time by a user. Since each LISP atom requires at least five memory cells, such an external "terms file" saves a considerable amount of space.

With respect to our use of TRAVELNET in the Semantics component of the speech system, we have written a procedure for taking the network, precomputing all allowable semantic association paths through it, and storing these paths, described by their type and endpoints, on an external file. This means that when one comes into the network at run-time with a new word or concept, trying to find what other words and concepts it can associate with, one doesn't have to load in and look at intermediate nodes. This technique results in savings in both time and space.

The range of understandable semantic associations used for noticing and proposing semantically related words and concepts was also extended this past year. Previously we could understand three types of associations among words and concepts: 1) associations based on being part of the same multi-word name (e.g., "computational" and "linguistics" in the name "computational linguistics"); (2) associations based on modification (e.g., "recent" and "meeting" in the phrase "recent meeting"); (3) associations based on case frames (e.g., "John" and "went" in "John went to the recent ACL meeting"). This set has now been extended to include firstname-lastname pairs, city-state and city-country pairs, and the general semantic notion of "property" (i.e., A is a property of B). Using this latter notion, we can now associate properties and the thing they are a property of, e.g., "location" (the property) and "conference" (e.g., "the

location of the conference", "the conference's location") or particular instances of a property and the thing it is a property of (e.g., "the Pittsburgh conference", "the conference in Pittsburgh").

Also with respect to TRAVELNET, the number of words and concepts described in TRAVELNET has grown to the point that 274 words from our 591 word dictionary have known semantic import to the system.

Finally, we incorporated this year a semantic interpreter into the SPEECHLIS system which transforms the parse trees output by the Syntax component into requests in our formal command language for execution in our data base. The semantic interpreter is very similar to the one used in the LUNAR system [Woods, Kaplan and Nash-Webber, 1972], with minor modifications to reflect differences in the output of the parser and the form of the command language. Its control structure is currently the same as that used in LUNAR, though we hope to improve upon that in the coming year to make the semantic interpreter more helpful with respect to error correction.

## I. User and Discourse Model

Early simulations of dialogues between a travel budget manager and his computer assistant suggested a discourse model in which, at any point, the user would be in one of

several states. He could be trying to determine the consequences of a proposed trip, or examining the state of the budget, or entering new trip information. During the past year we have considered several formulations of such a discourse model, on the assumption that state based expectations will be useful in understanding.

One such formulation has involved the concepts of modes of interaction and intents [Woods et al., 1974, pp. 201-232; Bruce, 1975a] An intent is the assumed purpose behind an utterance. Patterns of intents such as,

```
user-enter-new-information
     system-point-out-contradiction
          user-ask-question
          system-answer-question
     user-make-editing-change
system-confirm-change,
```

constitute the modes of interaction.

An augmented transition network (ATN) grammar has been used to represent some of the common modes of interaction found in travel budget management dialogues. A modified ATN parser has been written that steps through the grammar on the basis of the input sentence structure and the then-current state of the data base. At any given state the parser can predict the most likely next intent and hence such things as the mood and head of the next utterance.

Another formulation of the user/discourse model involves the notion of demands and counter-demands made by

participants in the dialogue. These include such things as unanswered questions and contradictions which have been pointed out. (Both of these formulations are discussed more fully in [Bruce, 1975b].

We are currently exploring mechanisms for applying discourse knowledge in parsing. One of these is the direct representation of pragmatic features in the grammar. Another is the use of discourse advice passed to syntax for use by functions which order the arcs in the grammar.

## J. Audio Response Generation

During the past year we have added audio response generation to the speech system. This has been done by coupling the text response generation program to the synthesis-by-rule program [see Section I.M.]. The programs allow the travel budget manager's assistant to speak to the manager in an English-like language. For example, it may describe a trip as:

> John Makhoul went to Pittsburgh from Monday, the 30th of June, to Wednesday, the 2nd of July, 1975.

This text string is then converted into a string of phonemes (obtained from the expanded phonemic dictionary) and passed to the synthesis program which produces a waveform file. Such a response generation capability also serves as a means of verifying phonemic spellings in the dictionary and testing the performance of the speech synthesizer-verifier.

(See Section II.D. for more details.)

## K. The Travel Budget Manager's Assistant

The current task domain of the BBN speech project is that of assisting a travel budget manager. The travel budget manager's assistant helps the manager to keep a record of trips taken or proposed and to produce summary information such as the total money allocated. The task is a simplified example of many other resource management problems of essentially the same type and is an initial step toward an intelligent manager's assistant.

During the past year we have run simulations to develop and circumscribe the travel budget manager's assistant. In the simulations, one person, sitting at terminal A, plays the role of the travel budget manager, typing in sentences as if he were talking to a complete travel budget manager's assistant. Another person, at terminal B, intercepts his sentences and translates them into a formal command language [Woods et al., 1975c, pp. 27-36], and passes the translations to the retrieval component for execution. These simulations also provide a source for dialog protocols and new words that should be included in the dictionary.

The current data base is implemented in the SEMNET formalism [Woods et al, 1975a, pp. 48-60]. Elements of the data base are items such as budgets, trips, contracts,

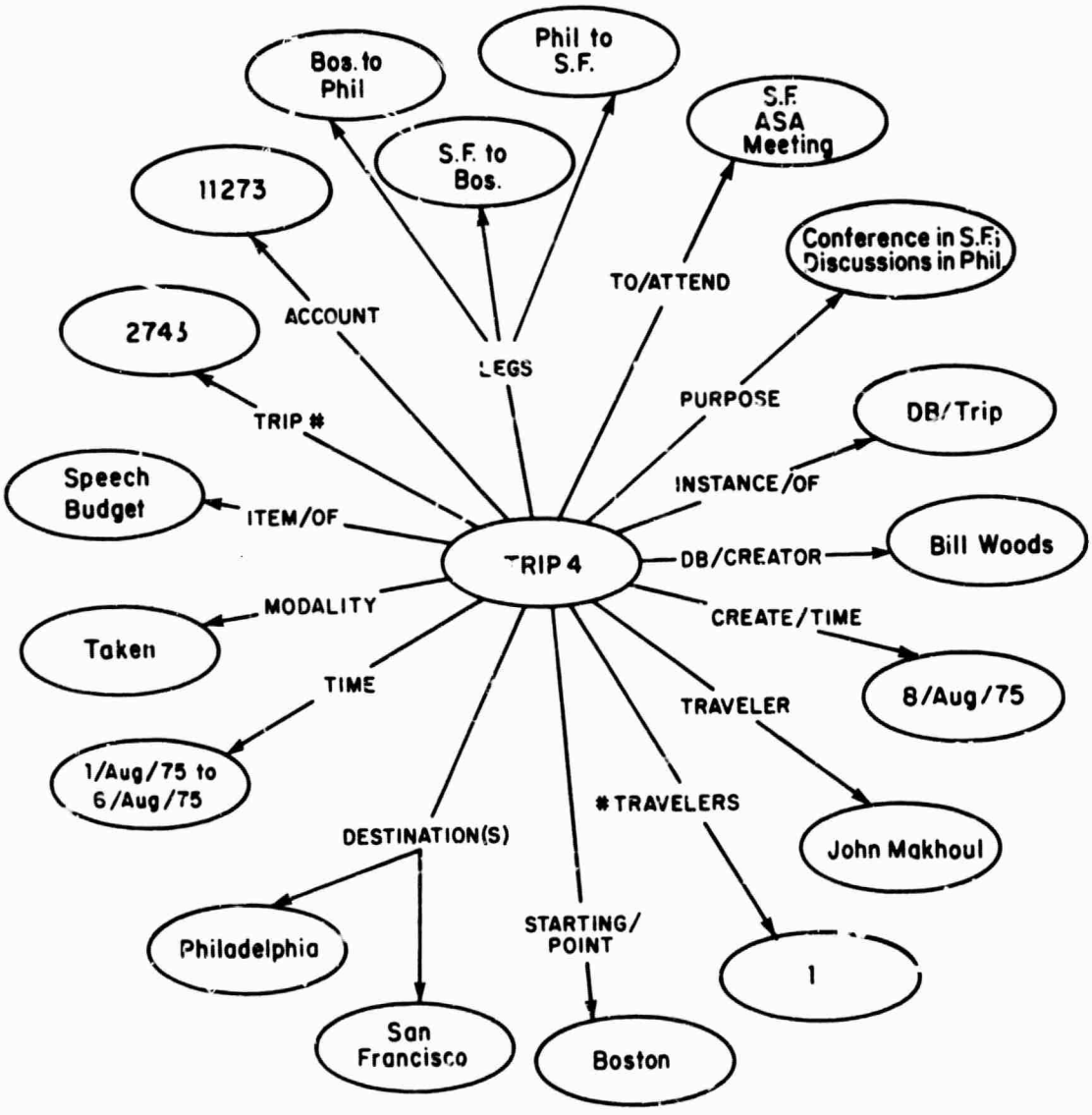


Fig. 4. Information stored in the travel budget manager's assistant data base for a single hypothetical trip.

conferences, fares, and dates [Woods et al., 1975c, pp. 15-26] Figure 4 shows most of the information which would be stored for a hypothetical trip taken by John Makhoul from Boston to Philadelphia and San Francisco. A future QTPR will discuss the actual details of the data base representations.

We are using "real" data taken from the speech research group's travel budget for the years 1975-1976 (currently about 40 taken and proposed trips). The network, which also contains information used by Semantics, has about 1200 nodes and is expected to double in size over the coming year.

The data base management facilities of the system are accessed through a formal command language [Woods et al., 1975c, pp. 27-36] into which typed or spoken requests can be translated. Inference in the system can be viewed as a natural generalization of the notion of structures with slots and default values for each slot. Here, instead of being values, defaults are procedures (METHOD's) for determining the appropriate value whenever a slot filler is missing. The procedures may in turn request other slot values, which may require activating other default procedures. The inference process also has an ultimate default, which is to ask the speaker. Similar procedures are used for adding new links (ADDFN's), adding new items (BUILDFN's), and establishing referents (REFFN's).

## L. Prosodics

Work on prosodics has been concerned with implementing and evaluating prosodics programs supplied by the UNIVAC speech understanding group. The first, BOUND3, a prosodic boundary detector, was tested here on 16 utterances by three speakers. Manually edited fundamental frequency contours were used due to inadequacies of the present pitch extraction routine, and the results were found to be roughly comparable to those reported by UNIVAC. We also ran a threshold-varying experiment on BOUND3 whose results showed that raising the rise and fall thresholds from 5 eighth-tones (7.5%) to 7 (10.6%) somewhat reduced the number of false boundaries found. In fact, the results for the 7 eighth-tone thresholds are nearly identical to the "after cutoff" results reported earlier for the recommended 5 eighth-tones threshold [Woods et al., 1975b, p. 33]. With respect to incorporating BOUND3 into a speech understanding system, we believe its present level of performance makes its use questionable. Its principal deficiencies lie in the number of false boundaries it detects and its uncertainty with respect to exact position of the ones found.

A second UNIVAC-supplied prosodics program, STRESS, has also been adapted to the BBN computing environment and data conventions, and both it and BOUND3 were made into subroutines callable from a budding prosodics component

named PROSOD. STRESS uses fundamental frequency, "sonorant energy" (energy in the 60-3000 Hz band), and the boundary results of BOUND3 to locate stressed syllables in the utterance. Further work on BOUND3 and STRESS has been suspended until a more reliable fundamental frequency extractor is completed (see Section I.A.).

A phase of increased interaction with the UNIVAC group was recently initiated on two fronts. Initially, they will be using the BBN-TENEXD system to carry on development of their prosodics programs, starting with the PROSOD/BOUND3/STRESS program described above. They will be operating on our on-line collection of utterances. The UNIVAC group will also be learning the details of the SPEECHLIS syntactic component so that they will be able to propose and test out specific algorithms for incorporating prosodic information into the operation of that component.

## M. Verification

During the past year, we were able to establish the Verification component as an operational part of the Speech Understanding System. In the current system configuration, it resides in a separate TENEX fork and communicates directly with the fork containing the Lexical Retrieval component. A special TENEX fork interface has been written to facilitate rapid transfer of data and control between

these two components.

In normal operation, Lexical Retrieval may select any word already matched at the phonetic level and request that its presence be verified at a particular location in the unknown utterance. The phonetic spelling of the matched word is given to the Verification component together with the frame numbers specifying the region of the utterance over which the match is to occur. The Verification component synthesizes a parametric representation of the word and matches it against the parameterization of the unknown utterance. A score indicating the likelihood of the word's presence is computed using a distance measure and a time normalization algorithm. This score augments that computed at the phonetic level by Lexical Retrieval.

The heart of the Verification component is a synthesis-by-rule program which computes a parametric representation for any word given its phonetic spelling. We have developed this program at BBN using a rule language called PCOMPILER [Woods et al., 1975b, pp. 18-23]. This allows us to specify the synthesis program's acoustic-phonetic and phonological components in a standard linguistic notation. PCOMPILER compiles them into efficiently operating Fortran code which supports the distinctive feature notation of the phones and the conditions for rule application. The synthesis program has

also been coupled with a terminal analog waveform synthesizer and used in our Audio Response Generation Component (see Section II.D.).

The Verification component also includes a time normalization routine and a parametric matching program. Time normalization is done using a dynamic programming algorithm based on a method first developed by Itakura [1975]. The algorithm involves a non-linear time warping based on the registration of the error metric, in this case, the ratio of the linear prediction residuals. We have modified Itakura's method to allow limited misalignment in time between the hypothesized word parameterization and that portion of an unknown utterance onto which we wish to match. In actually computing the "distance" between a hypothesized word and a portion of the unknown utterance, we sum the logarithms of the ratio of the prediction residuals between corresponding segments, the correspondence having already been determined by the time normalization technique. Comparing the linear prediction residuals is a method of spectral matching, specifically the spectra of the all-pole models.

In exercising the Verification component, we have observed its usefulness in cases where segmentation or labeling errors by the APR have caused Lexical Retrieval to give a prospective word a low score. The ability to verify

a sequence of words has also been useful since a segmentation error can cause Lexical Retrieval to overlap the highest scoring occurrences of two adjacent words and prevent their being proposed sequentially. We are now studying how best to integrate these capabilities into control strategies being developed for the entire SPEECHLIS system.

## N. Recognition Strategies

In the past year, a set of primitive operations was developed for directing the Syntactic, Semantic and Lexical Retrieval components and manipulating relevant data structures in order to arrive at a plausible model of an input utterance [Woods et al., 1975c, pp. 37-50]. Using our experience with incremental simulations [Woods and Makhoul, 1973], we have recently begun to construct various recognition strategies out of these primitives and to try them out on the recorded utterances in our acoustic data base. So far, three strategies have emerged which differ in the predictive and evaluative power they expect from the three components. The first strategy relies initially on a strong Semantic component and then on a Syntactic component which can be strongly predictive when given a long enough string of wordmatches to work with. The second strategy expects a powerful Lexical Retrieval component, and the third, a strongly predictive Syntactic component. The

strategies succeed or fail then on the validity of these assumptions for the existing Syntactic, Semantic and Lexical Retrieval components.

In the first strategy, we attempt to reconstruct the utterance in a mainly left to right mode. We scan for good wordmatches anchored at the beginning of the utterance, and then make a theory for the best one. That theory is sent to Semantics which may make proposals based on it. Any proposals are then done, followed by any events resulting from the proposals. We continue left-to-right, always doing anchored scans, until there are neither proposals nor events left. Syntax is then called upon to judge the acceptibility of the current theory and to try to extend it, based on what it might expect next. The only diversion from strict left-to-right search occurs when the best matching word found on a anchored scan is long (i.e., >5 phonemes in length). In that case, since a better match might be found without the anchoring, we make a sliding scan for the same word. Syntax is then used to fill in the small gap that might result. Using this strategy, we were able to recognize two utterances out of three tried.

This first strategy was designed to be used with the BBN SPEECHGRAMMAR (see I.G.). This grammar accepts a very large variety of syntactic constructions. Consequently, it is difficult for Syntax to tell if a string of wordmatches

can or cannot be extended into a valid construction, or if it can, in which one. This limits its ability as either a judge or a predictor, unless the string it is given is long enough to considerably limit the possibilities. This is the reason for not evoking Syntax until so late in the strategy. When Syntax is used with SMALLGRAM, the more constrained grammar (see I.G.), it becomes much more powerful with respect to its ability to evaluate theories and make predictions. While we have not yet done so, it will be edifying to try a left-to-right strategy using SMALLGRAM and see what it buys us in terms of speed and success rate.

In the second strategy, a wordmatch "cover" of the utterance is established using the Lexical Retrieval component alone. This is done by choosing as part of the cover the highest scoring wordmatch returned on a scan for the best fifteen. This leaves two empty regions in the utterance, or perhaps one if the best wordmatch abuts one end. A scan is then made for the best fifteen matches in each empty region. Picking the best match in each region may close it up or result in new, but smaller, empty regions. Successive anchoring and scanning will result in a cover for the utterance, plus many other good wordmatches. The other matches are collected into "conflict sets", one for each member of the cover. The wordmatches in the conflict set for wordmatch X are competitors for some region of the utterance covered by X. The conflict sets are then

trimmed to include only those wordmatches whose score is reasonable close to that of the relevant cover wordmatch, X. The result of this lexical retrieval phase is a cover of the utterance with good wordmatches and the conflict set associated with each. The critical assumption is that a high proportion of the words actually in the utterance will be picked up as part of the cover and that the ones which are not will be found in some conflict set.

After the lexical matching phase, Semantics is called in to look for clusters of wordmatches which associate semantically with each other. It is hypothesized that those words in the cover which are also in a large semantic cluster are part of the original utterance. These words are then given to Syntax to flesh out if possible into a complete spanning theory, with their conflict sets providing alternatives for use in case Syntax fails. Again this second strategy was designed to be used with SPEECHGRAMMAR for the same reasons as the first strategy.

While the strategy has never been exercised to completion on an utterance due to space problems during the semantic association phase, we did go so far as to produce conflict sets for all the utterances in the speech data base. This has proved invaluable in locating deficiencies and errors in the APR and Lexical Retrieval components. As a result, the production of cover and conflict sets for the

data base utterances has become part of the evaluation procedure for new versions of the APR and Lexical Retrieval components. (Should we decide to experiment further with this strategy in its current form, we can overcome the space problem by putting the Semantic Association component in a separate fork, rather than with the Speech recognition programs.)

In the third strategy, the system is driven by the Syntactic component trying to find a grammatical model of the utterance. In order that the possible combinatorics not make this strategy impossible, we have run it using SMALLGRAM, the more constrained and therefore more predictive of the two SPEECHLIS grammars (see I.G.). In this strategy, a preliminary scan is made for the 15 best wordmatches. Wordmatches are taken in order of lexical score as the seeds for a syntactic theory, until one can be extended into a theory which spans the utterance. Given a seed theory, Syntax attempts to grow it out to the left and the right, using a combination of notices and proposals. To prime the pump for Syntax and enable it to notice desirable adjacent words without having to propose them, the Lexical Retrieval component may be asked at each stage to scan off the sides of the theory being grown by Syntax to find the best matching words at either end. We have so far tried this strategy on 14 utterances using the segment lattices produced by the Spring 1975 version of the APR and 8

utterances using the improved lattices produced by the new version. In the first group, three utterances were recognized automatically, and in the second group, four utterances. In the coming year, we will be working on extending and improving this strategy, noticing the effect on its operation brought about by extensions and additions to SMALLGRAM.

## 0. PDP11/SPS41 Signal Processing System

During the past two years we have been gathering the components for a special purpose signal processing system. The essential components of this system are:

1. DEC PDP11/40, including 32K core memory, extended instruction set.
2. Standard Memories 24K core memory.
3. Telefile DC-16H/Century CD213 disc, 30 million words.
4. SPS-41 signal processor, including 8K semiconductor shared memory and dual A/D and D/A converters.
5. IMP-11A ARPANET interface.
6. ELF operating system, including virtual memory, user programs, and file system.

Items 2 and 3 were delivered during the previous contract year. The remaining items were delivered (or released, in the case of ELF) during the present contract year, although items 5 and 6 became operable only in the late summer. Documentation of ELF is presently fair-to-poor, but

apparently progressing.

The lack of reliability of the SPS-41 has been a disappointment. For example, although it would run the SPS-supplied diagnostic programs, it would hang up on the Network Speech Compression project's LPC vocoder program. As part of an NSC effort to find out what is needed to make the SPS-41 run the vocoder program our machine was sent back to SPS Inc. in late August. Sufficient improvements were made to the machine to get it to the state where it would run the vocoder program, but only on some occasions. Its overall reliability is still in question. It is due to be shipped back to BBN in the latter part of November.

The combination of late availability of essential system components and the SPS-41 reliability problems has severely impeded progress in turning these individual components into a signal processing system. We expect to be able to do this in the coming contract year.

## References

Bruce, B. (1975a)
"Pragmatics in Speech Understanding," Proc. Fourth International Joint Conference on Artificial Intelligence, Tbilisi.

Bruce, B. (1975b)
"Discourse Models and Language Comprehension," Thirteenth Annual Association for Computational Linguistics Meeting, Boston. To appear in the American Journal for Computational Linguistics.

Gillmann, R. A. (1975)
"A Fast Frequency Domain Pitch Algorithm," J. Acoust. Soc. Amer. 58 Supplement 1, p. S63 (presented at the 90th meeting of the ASA, 5 Nov. 1975, San Francisco).

Itakura, F., and S. Saito (1971)
"Digital Filtering Techniques for Speech Analysis and Synthesis," 7th Int. Cong. Acoust., Budapest, 25 C 1.

Itakura, F. (1975)
"Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. on Audio, Speech, and Signal Processing, Vol. ASSP-23, pp. 67-72.

Newell, A. et al. (1973)
Speech-Understanding Systems: Final Report of a Study Group, North-Holland/American Elsevier.

Rabiner, L. R., and M. R. Sambur (1975)
"An Algorithm for Determining the Endpoints of Isolated Utterances," Bell System Tech. J. 54, pp. 297-315.

Schwartz, R. M."(1971)
"Automatic Normalization for Recognition of Vowels of All Speakers," S.B. Thesis, MIT, Cambridge, Mass.

Wakita, H. (1975)
"An Approach to Vowel Normalization," J. Acoust. Soc. Amer. 57 Supplement 1, p. S3 (presented at the 89th meeting of the ASA, 8 Apr. 1975, Austin).

Woods, W.A., R. Kaplan and B. Nash-Webber (1972)
"The Lunar Sciences Natural Language Information System: Final Report," Report No. 2378, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W.A. and J. Makhoul (1973)
"Mechanical Inference Problems in Continuous Speech Understanding," Proc. Third International Joint Conference on Artificial Intelligence, pp. 200-207. (Reprinted in Artificial Intelligence, Vol. 5, No. 1, PP. 73-91, Spring 1974).

Woods, W. A., M. Bates, B. Bruce, J. Colarusso, C. Cook, L. Gould, D. Grabel, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf (1974)
"Natural Communications with Computers Final Report - Volume I. Speech Understanding Research at BBN, October 1970 to December 1974," Report No. 2976, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W. A., R. Schwartz, J. Klovstad, C. Cook, J. Wolf, M. Bates, B. Nash-Webber, B. Bruce, V. Zue (1975a)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 1, 1 November 1974 to 1 February 1975," Report No. 3018, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W.A., R. Schwartz, C. Cook, D. Klatt, J. Wolf, M. Bates, B. Nash-Webber, B. Bruce, and J. Makhoul (1975b)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 2, 1 February 1975 to 1 May 1975," Report No. 3080, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W.A., R. Schwartz, C. Cook, J. Klovstad, M. Bates, B. Nash- Webber, B. Bruce, and J. Makhoul (1975c)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 3, 1 May 1975 to 1 August 1975," Report No. 3115, Bolt Beranek and Newman Inc., Cambridge, Ma.

Zue, V. and A. Paige (1970)
"Computation of Vocal Tract Area Functions," IEEE Trans. Audio Electroacoust., AU-18, pp. 7-18

# II. TECHNICAL NOTES

## II.A. Parametric Modeling of Probability Distributions

John Makhoul
Richard Schwartz

In designing our APR and lexical retrieval components, we have explicitly allowed for a labeling strategy where each segment is characterized probabilistically [Woods et al., 1974; Woods et al., 1975]. Thus, instead of associating a single phoneme with a particular segment of speech, we now label each segment by a vector of phonemes with a corresponding vector of probabilities. For each phoneme, the probability is computed as the likelihood that the given segment is indeed that phoneme. At present, many of these probabilities are taken from long-term confusions with the first choice phoneme, which are averaged over many variables and are therefore very approximate.

In order to render these computations more rigorous and accurate, we need to estimate the actual probability distributions for different acoustic events as a function of the pertinent acoustic parameters. These distributions could be estimated (as histograms, for example) from our data base using our existing statistics package [Woods et al., 1974; Schwartz, 1975]. However, such a nonparametric estimation of distribution is cumbersome to use and costly in terms of storage. It would be more convenient to have a

parametric representation in which each distribution is represented by a small number of parameters. Two techniques for such parametric modeling of one-dimensional probability distributions have been developed and implemented here, the Beta distribution and linear prediction, both of which are also extendable to multidimensional distributions. These two methods of parametric modeling share the property that they are defined over a finite interval. Many of the traditional parametric distributions (such as the Gaussian distribution) are defined over the infinite real line. However, such models do not generally allow for skewness in the distribution. This limitation is overcome in the Beta and linear prediction methods.

## Beta Method

Of the many parametric probability distributions available in the literature, the Beta distribution (with a number of related distributions [Elderton, 1969]) stands unique in that it is defined over a finite interval and allows for skewness in the distribution. The probability density is given by

$$p(x) = \begin{cases} A(a,b)\ x^a\ (1-x)^b, & 0 \le x \le 1,\ a,b \ge 0, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $$A(a,b) = \frac{\Gamma(a+b+2)}{\Gamma(a+1)\ \Gamma(b+1)}, \tag{2}$$

and $\Gamma(\cdot)$ is the Gamma (factorial) function. This distribution is unimodal with a maximum at

$$x|_{max} = \frac{a}{a+b} \quad . \tag{3}$$

The mean and variance are given by

$$m = \frac{a+1}{a+b+2} \tag{4}$$

and

$$\sigma^2 = \frac{(a+1)\ (b+1)}{(a+b+2)^2\ (a+b+3)} \quad . \tag{5}$$

Other properties include

$$p(x) = 0, \quad x = 0 \text{ and } x = 1 \tag{6}$$

$$\frac{dp(x)}{dx} = \begin{cases} 0, & x=0,\ a\geq i, \quad \text{or} \quad x=1,\ b\geq 1, \\ \infty, & x=0,\ a<1, \quad \text{or} \quad x=1,\ b<1. \end{cases} \tag{7}$$

For a=b, the density function is symmetric about x=0.5. For a>b, it is skewed to the right, and for a<b, it is skewed to the left.

Let us assume that for some acoustic feature or phoneme the sample values of some acoustic parameter (variable) x are $x_i$, $1\leq i\leq N$. The problem is how to model the desired distribution by a Beta distribution. The method we have adopted is to set the mean and variance of the desired Beta distribution to be equal to the sample mean and variance of the data. However, first we must define the range of x and normalize it to [0,1]. Let

$x_o$ = minimum x for the model distribution,

$x_m$ = maximum x for the model distribution.

We choose $x_o$ and $x_m$ such that

$$x_o < x_i < x_m \quad , \quad 1 \leq i \leq N . \tag{8}$$

(Note that the set $\{x_i\}$ need not be ordered, i.e., $x_2 > x_1$ need not hold, etc.) The given variable x is normalized by defining a new variable y given by

$$y = \frac{x - x_o}{x_m - x_o} \quad . \tag{9}$$

It is clear that y varies between 0 and 1. Now, let the sample mean and variance of $\{x_i\}$ be given by

$$m_x = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{10}$$

$$\sigma_x^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - m_x)^2 = \frac{1}{N} \sum_{i=1}^{N} x_i^2 - m_x^2 \; . \tag{11}$$

Then, one can show that the mean and variance for the normalized variable y are given by

$$m_y = \frac{m_x - x_o}{x_m - x_o} \tag{12}$$

$$\sigma_y^2 = \frac{\sigma_x^2}{(x_m - x_o)^2} \quad . \tag{13}$$

We now determine the values of the parameters a and b of the Beta distribution by equating (12) to (4) and (13) to (5).

The result is:

$$a = \frac{m_y^2(1-m_y)}{\sigma_y^2} - m_y - 1 \qquad (14a)$$

$$b = \frac{a+1}{m_y} - a - 2 = \frac{m_y(1-m_y)^2}{\sigma_y^2} + m_y - 2. \qquad (14b)$$

This completes the specification of the Beta distribution that models the desired probability distribution.

It is important to note that no error criterion was used in the modeling process, so some problems are expected to arise. One such problem is that one or both of the values of a and b computed from (14) may come out negative, which is not allowed by (1). A negative power in (1) means that the density goes to infinity at either 0 or 1; a situation that we feel is undesirable for our modeling purposes. What this indicates is that the model is not a good fit to the data. This situation can occur if N is small and the samples are wide apart, or if the distribution is multimodal. (If the few samples are widely separated, they look like a multimodal distribution.) Clearly, the Beta distribution is not meant to handle such situations.

The solution we have adopted is to choose $x_m$ and $x_o$ such that the distance $(x_m - x_o)$ is increased. From (13) we see that $\sigma_y^2$ is thereby decreased, and from (14), that a and b are correspondingly increased (for a fixed $m_y$). It might seem from (12) that $m_y$ would also decrease. However, note

that $m_y$ depends on the actual value of $x_o$ as well, which can be adjusted so as not to decrease $m_y$. In general, one can show that it is always possible to ensure the positiveness of a and b for a well-chosen $x_o$ and $x_m$. If $(x_m-x_o)$ is made too large, however, the effect is to have large values for a and b, which makes for a very narrow distribution that might not be a good approximation to the actual distribution.

Taking these problems into account, we therefore use the following procedure for computing a Beta distribution parametric model:

1. Compute the sample mean and variance from (10) and (11).
2. Choose $x_o$ and $x_m$ to be outside the range of the data $\{x_i\}$. (A few percent of the range is usually sufficient.)
3. Compute the mean and variance of the normalized variable y from (12) and (13).
4. Compute the parameters of the Beta distribution, a and b, from (14).
5. If $a \geq 0$ and $b \geq 0$, done.
6. Otherwise, decrease $x_o$ and increase $x_m$ a little.
7. Go to (3) above.

It is best to use a display that shows the computed Beta distribution superimposed over a histogram or other nonparametric estimate of the distribution. The experimenter can then choose $x_o$ and $x_m$ more judiciously.

Linear Prediction Method

The Beta method has the limitation that it produces a unimodal distribution, and that there is no error criterion associated with the modeling process. However, the linear prediction (LP) method to be described does not have these limitations. The resulting model distribution is not necessarily unimodal and is computed through the minimization of a well defined error criterion.

This method borrows from the theory of linear prediction where a signal spectrum is modeled by an all-pole (autoregressive) model spectrum [Makhoul, 1975a and 1975b]. Since the only condition for this theory to apply is that the function (spectrum in that case) be positive definite, the same theory can apply to the modeling of probability densities, which are also positive definite. However, there remains the question of whether the error criterion used in LP modeling is also appropriate for the modeling of probability densities. This is discussed below.

In LP theory, a positive definite function (spectrum) p(x), defined over the domain [0,1], is modeled by an all-pole spectrum q(x) given by

$$q(x) = \frac{G_M}{\left|1 + \sum_{k=1}^{M} a_k \, e^{-j\pi x}\right|^2} \qquad (15)$$

where $a_k$ are known as the predictor coefficients, M is the number of poles in the spectrum, and $G_M$ is a normalizing constant. The coefficients $a_k$ are obtained as a result of minimizing an error criterion E, where

$$E = \int_0^1 \frac{p(x)}{q(x)}\, dx \ . \tag{16}$$

Substituting (15) in (16) and minimizing E with respect to $a_k$, $1 \le k \le M$, one obtains

$$\sum_{k=1}^{M} a_k\, r(i-k) = -r(i), \quad 1 \le i \le M, \tag{17}$$

where

$$r(i) = \int_0^1 p(x) \cos(\pi x)\, dx \tag{18}$$

is the autocorrelation function. (For a probability density, r(i) is the characteristic function.) (17) is a set of M linear equations in M unknowns, which can be easily solved for the predictor coefficients. The normalizing factor $G_M$ is obtained by setting the integral of q(x) to 1. The answer can be shown to be equal to the minimum of E, $E_M$, which is given by

$$G_M = E_M = 1 + \sum_{k=1}^{M} a_k\, r(k) \ . \tag{19}$$

(We assume in (19) that the integral of p(x) is also 1.)

The error criterion in (16) is useful for spectral modeling because one is usually interested in relative power

as opposed to absolute power. For modeling of probability densities it is not always clear which is to be preferred. However, if one is likely to make heavy use of likelihoods, which employ ratios of densities, then (16) is an appropriate error measure.

Now, given a set of sample measurements, $x_i$, $1 \leq i \leq N$, we wish to model the probability density of x by an all-pole function given by (15). First, one must normalize the sample data to the domain [0,1]. This is done by defining a normalized variable y given by (9). Our sample density function p(y) is now given by

$$p(y) = \frac{1}{N} \sum_{i=1}^{N} \delta(y-y_i), \tag{20}$$

where $\delta(x)$ is the Dirac delta function, and $y_i$ are computed from (9). Substituting (20) in (18), we have for the characteristic function

$$r(i) = \frac{1}{N} \sum_{i=1}^{N} \cos(\pi y_i) \ . \tag{21}$$

Now, we solve (17) and substitute in (15). The whole procedure is then, as follows:

1. Choose $x_o$ and $x_m$ outside the range of $\{x_i\}$.
2. Normalize the samples by substituting $x_i$ for x in (9).
3. Compute r(i) from (21) for $1 \leq i \leq M$.
4. Solve for $a_k$, $1 \leq k \leq M$, from (17).
5. Compute $G_M$ from (19).

q(x) in (15) is then completely specified.

There remains the problem of determining an "optimal" value for M. This is discussed in [Makhoul, 1975a] for the case of spectral modeling. For most of the densities we have modeled, M was always between 2 and 4. As M is increased, p(x) will have more peaks, introducing the possibility of modeling multimodal densities. As $M \to \infty$, q(x) will be identical to p(x), which is the degenerate case of an N-modal density, one mode at each of the samples $x_i$.

There is one disadvantage to LP modeling, and that is a property of the model density q(x). For finite M, q(x) can never be zero at any value of x. That could result in modeling errors, especially at x=0 and x=1. We mention in passing that the slope of q(x) is zero at x=0 and x=1.

Examples

Figs. 1-4 show examples of modeling the probability density of minimum ROP (energy of the preemphasized signal) for certain phonetic environments. The number of samples in this case was N=84. In each of the figures, the model density is plotted against a histogram of the sample data to show the rough location of the samples.

Fig. 1 shows a Beta density fit to the data, with $x_o$ and $x_m$ set outside the minimum and maximum values of the

samples, respectively, by an amount equal to 5% of the range of the sample values. Fig. 2 shows another Beta density fit with the same $x_m$ but with $x_o$ lowered. Note the displacement of the peak of the Beta density and the change in the values of a and b. In particular, notice that the value of a changed from being less than 1 in Fig. 1 to being greater than 1 in Fig. 2, with a corresponding change in the slope at $x_o$ (see (7)).

Figs. 3 and 4 show LP fits to the data for the same conditions as in Figs. 1 and 2, respectively. The number of poles in each case is M=4. Note how the value and location of the peak is quite dependent on the value of $x_o$ in this example. A similar shift in $x_m$ would not have produced the same effect for this example. In comparing the LP fits to the Beta fits, it is clear that because of the larger number of degrees of freedom in the LP model (4 against 2 in this example), one has greater freedom in selecting the model that best fits the data.

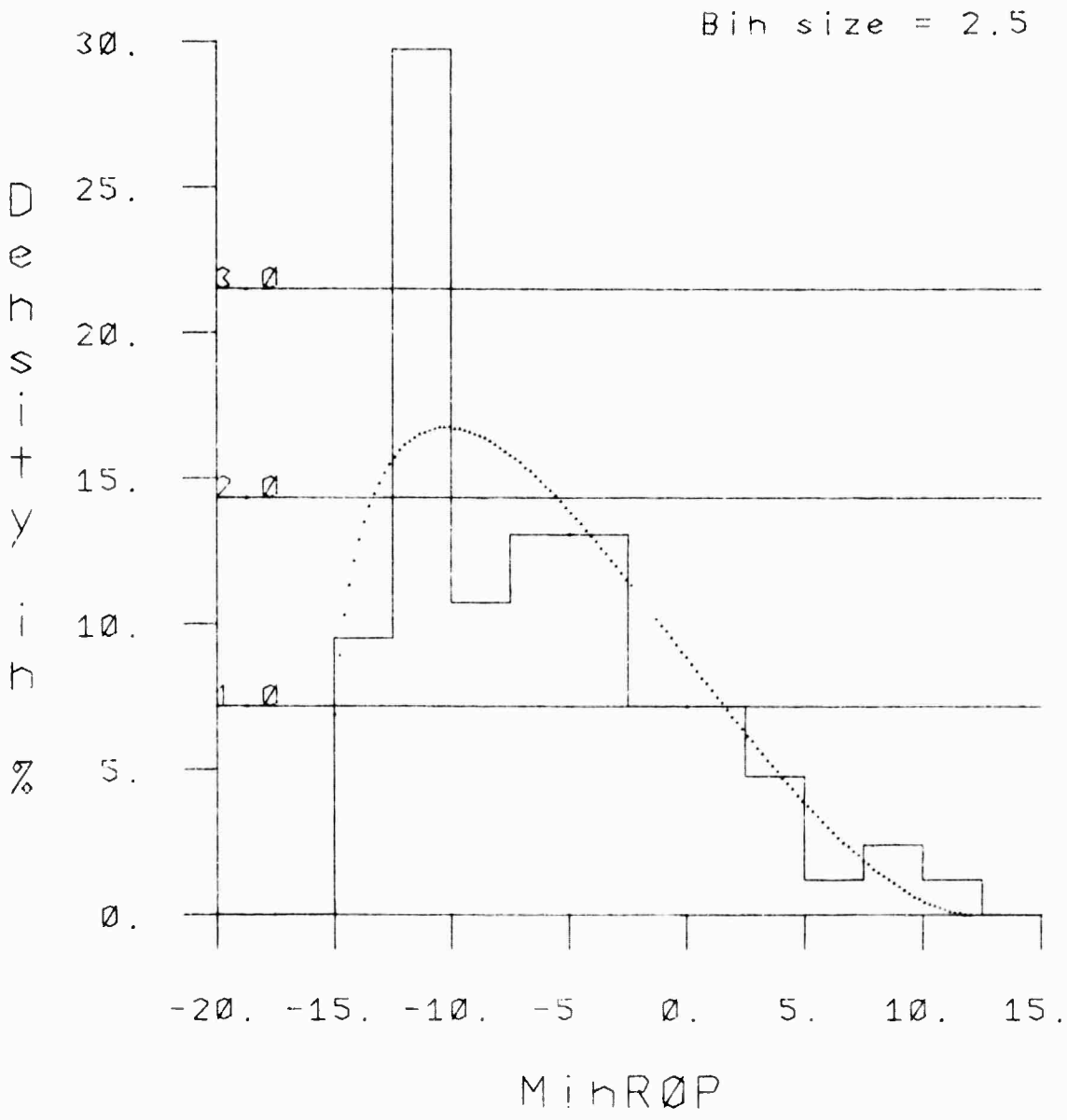


Fig. 1. A Beta density fit with a=.39 and b=1.74.
$x_o$ and $x_m$ set 5% outside the range of the data.

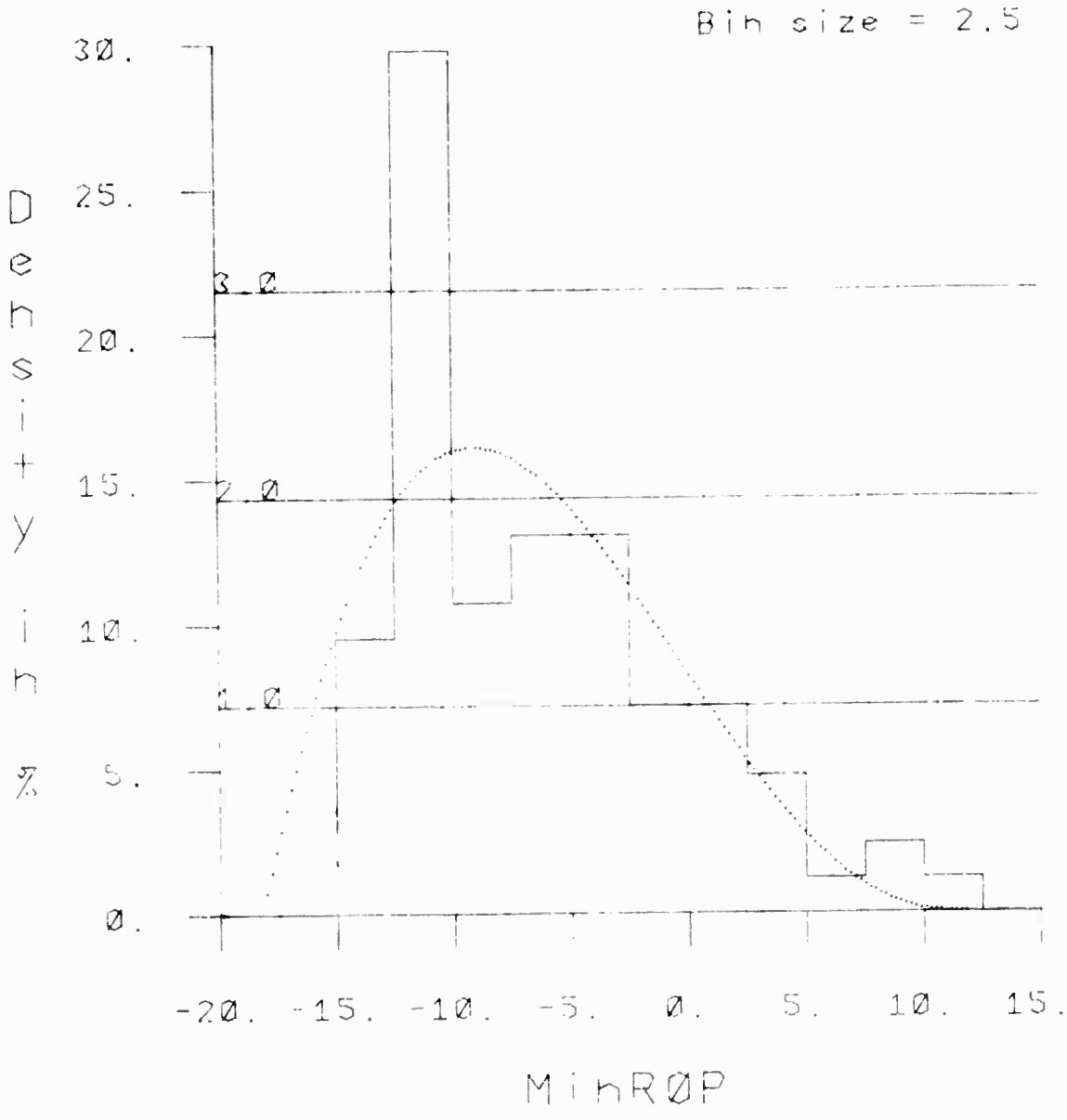


Fig. 2. A Beta density fit with a = 1.07 and b = 2.47. Same $x_m$, but a lower $x_o$.

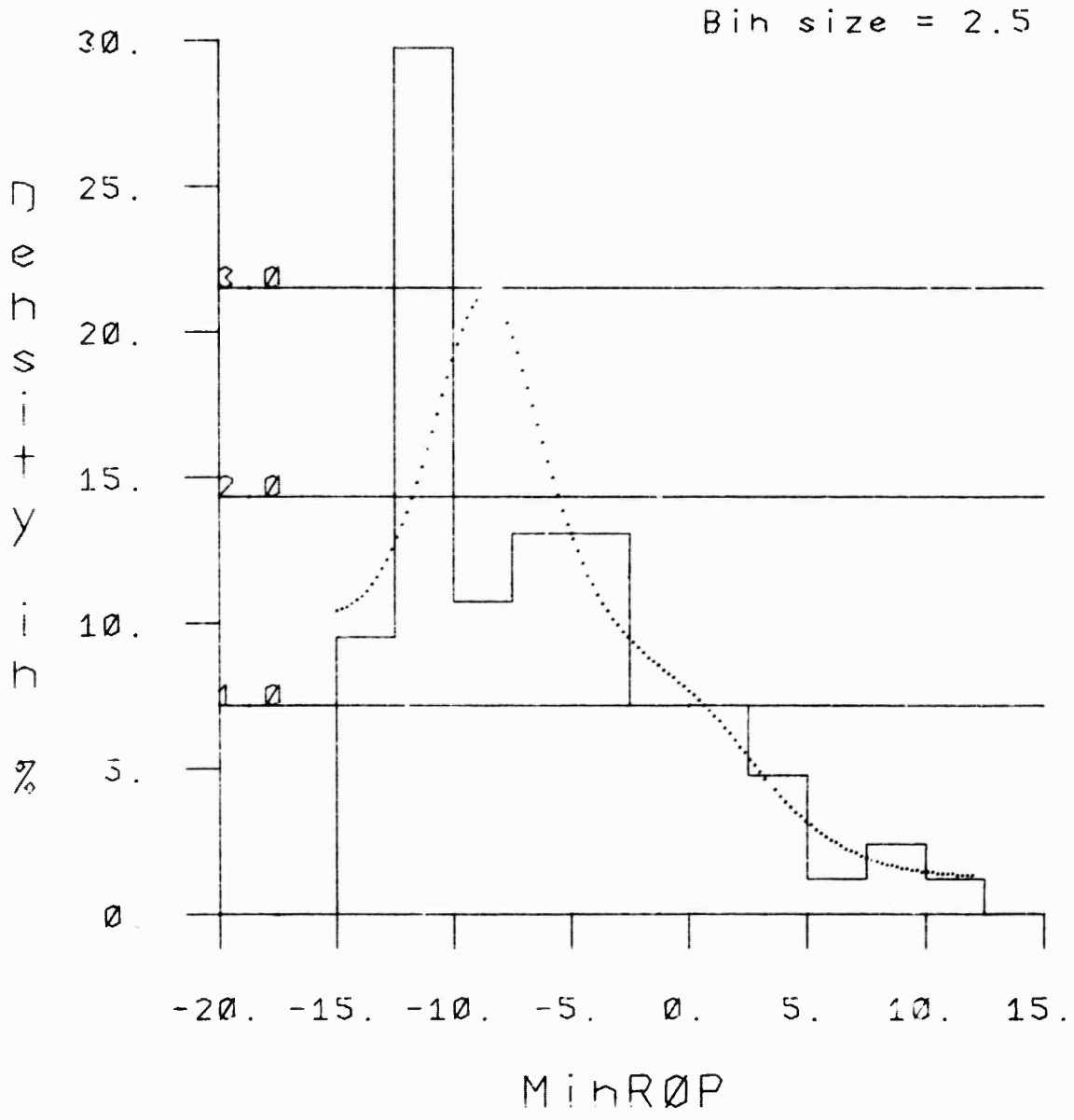


Fig. 3. A linear prediction fit with M = 4. $x_o$ and $x_m$ same as in Fig. 1

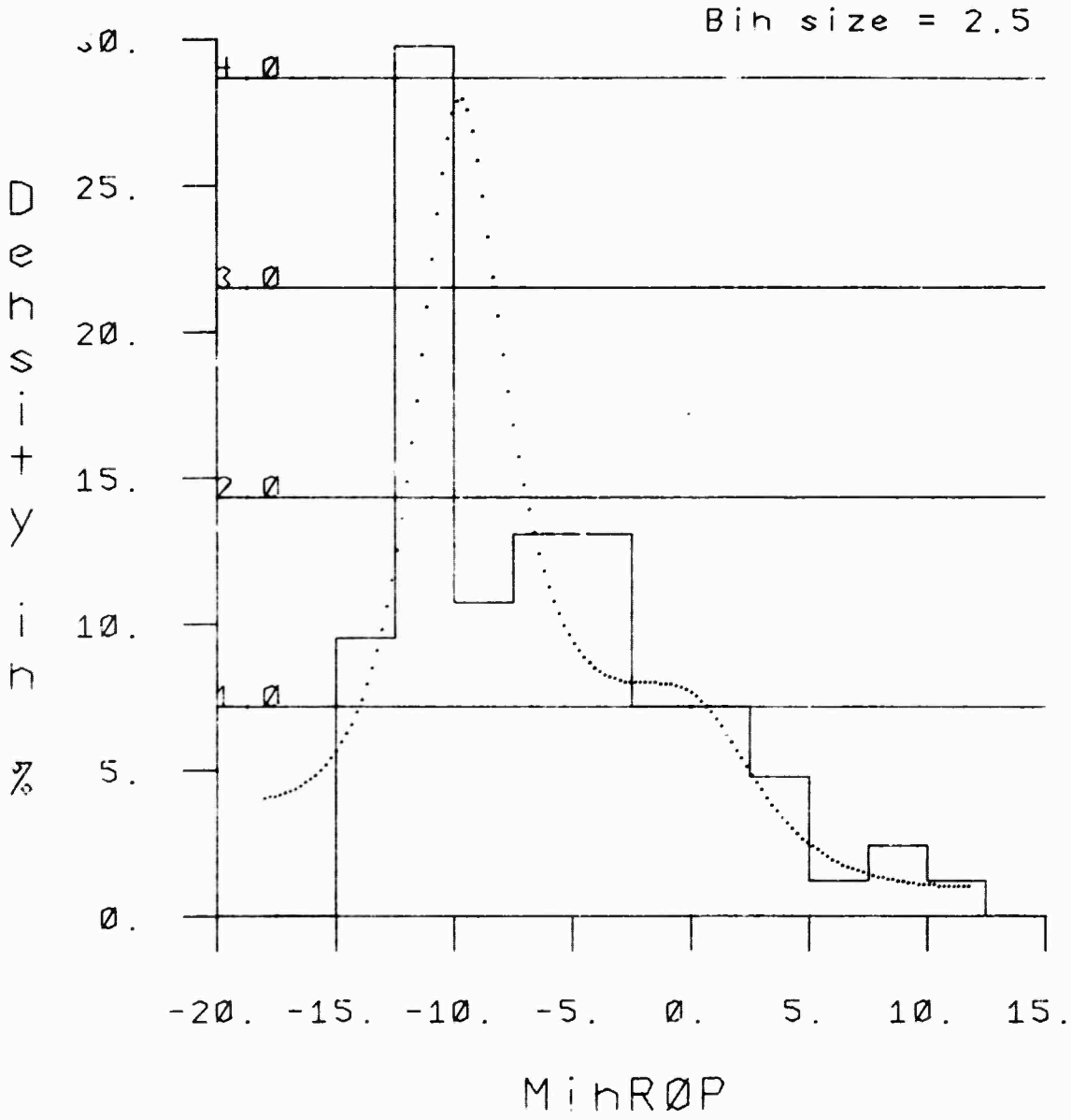


Fig. 4. A linear prediction fit with M = 4. $x_o$ and $x_m$ same as in Fig. 2.

## References

Elderton, W. P. and N. L. Johnson (1969)
Systems of Frequency Curves, Cambridge University Press, Great Britain.

Makhoul, J. (1975a)
"Linear Prediction: A Tutorial Review," invited paper, IEEE Proceedings special issue on Digital Signal Processing, Vol. 63, No. 4. 561-580, April.

Makhoul, J. (1975b)
"Spectral Linear Prediction: Properties and Applications," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-23, No. 3, 283-296, June.

Schwartz, R. M. (1975)
"Acoustic-phonetic Experiment Facility for the Study of Continuous Speech," J. Acoust. Soc. Am., Vol. 58, Supplement 1, 105, Fall.

Woods, W. A., M. Bates, B. Bruce, J. Colarussc, C. Cook, L. Gould , D. Grabel, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf (1974)
"Natural Communications with Computers, Final Report - Vol. I: Speech Understanding Resarch at BBN, October 1970 to December 1974," Report No. 2976, Vol. I, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W.A., R. M. Schwartz, J. Klovstad, C. Cook, J. Wolf, M. Bates, B. Nash-Webber, B. Bruce, and V. Zue (1975)
"Speech Understanding Systems, QPR No. 1, 1 November 1974 to 1 February 1975," Report No. 3018, Bolt Beranek and Newman Inc., Cambridge, Ma.

## B. Digital Spectrograms

Craig Cook

It is frequently desirable to examine directly the result of a spectral modeling algorithm across an expanse of time without going through an analog tape to a conventional spectrograph. To this end, we have written a program which computes good quality digital spectrograms and allows them to be displayed on a high-resolution electrostatic printer (in this case, a Xerox Graphics Printer). Given digital input of a 20 msec window on a 10 msec frame, the output display is comparable in quality to that of an analog spectrograph. In addition, it has the advantage that the ratio of the time and frequency axes (aspect ratio) can be varied, as can the frequency range to be displayed.

Figure 1 shows the output of the program for a preemphasized LPC spectrum (0-5KHz) approximately 2.3 seconds in length. The program computes a spectral shape for every 10 msecs in time and generates the display by choosing the appropriate one of 16 gray levels for each of 128 spectral intensity points along the frequency axis (1 point every 39 Hz).

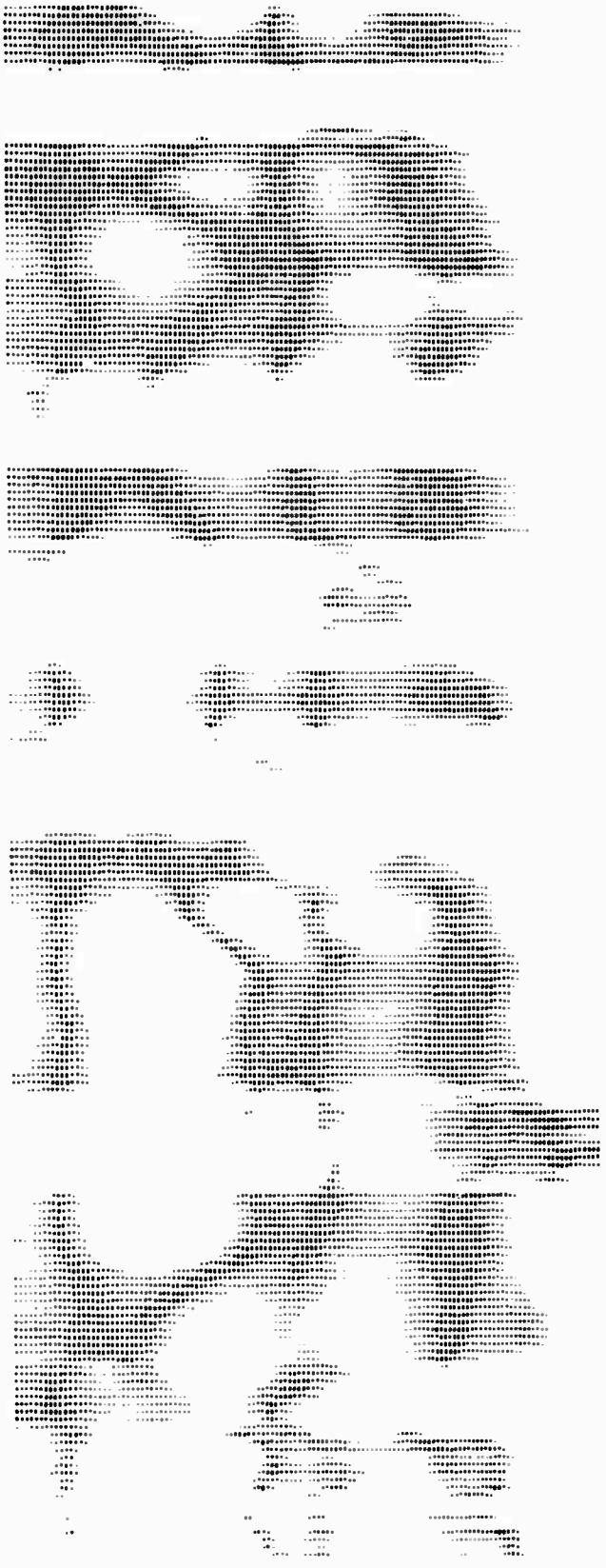

Fig. 1. (.9 actual size).

## C. A Phonological Rule System for Dictionary Expansion

W. A. Woods

### 1. Introduction

In their 1968 ACM paper, Bobrow and Fraser presented a set of notations for expressing phonological rules in a machine readable form and an interactive system for testing them [Bobrow and Fraser, 1968]. In 1973, a version of this system was brought into use at Speech Communications Research Laboratory in Santa Barbara as part of the ARPA speech understanding program (Brill and Hayden, Network message, 15-DEC-73). The program allowed a user to test a sequence of obligatory rules against a string of phonetic elements, with the possibility of tracing the intermediate steps of the operation of the rules. It did not however, provide facilities for applying all permissible sequences of optional rules to a single word, nor a facility for applying the rules systematically to all of the words in a vocabulary and producing tabulations of the results.

This paper presents a set of modifications to the Bobrow-Fraser rule tester which supply such features, providing a system for phonological rule debugging and rule-driven dictionary expansion for use in a speech understanding system. The system is implemented in INTERLISP as a set of supplementary routines to the

Bobrow-Fraser program, and is operational at Bolt Beranek and Newman Inc. where it is being used to expand the base-form dictionary for the BBN speech understanding sytem.

## 2. Features of the Bobrow-Fraser Rule Notation

The rule tester permits one to define a set of phonemes as bundles of features a la the Chomsky-Halle models in "The Sound Patterns of English" [Chomsky and Halle, 1968]. This is done by putting a list of the feature names for which the phoneme has positive values on the property list of the phoneme under the property name PHONEME. A facility for doing this and also keeping track of the set of possible phoneme features is provided in the Bobrow-Fraser system by a command DP which is used as in the following example:

```
DP P (CONS OBST LAB ANT)
```

This defines the phoneme P as +consonant, +obstruent, +labial, and +anterior. The phoneme is implicitly defined as -X for all other features X (including ones that would normally be considered relevant to only certain classes of phonemes). The command also adds P to the list PHONEMES which remembers all of the phonemes that have been defined.

In addition, the rule tester permits one to define a set of phonological rules in the traditional left and right context form using the symbol / to separate the rule proper from the context required for its application and -- to

indicate the position in the context where the rule applies. The operator for defining a phonological rule is DR, which is used as in the following example:

```
DR Z4A (EY EH / # (# 0 3 (+ CONS)) -- !0 (# 0 2 (+ CONS)) #)
```

This defines the rule Z4A as a rule that replaces the phoneme EY (as in bait) with the phoneme EH (as in bet) in the context consisting of a word boundary (#) followed by from 0 to 3 segments marked +consonant on the left, and a zero stress mark (!0) followed by 0 to 2 consonants followed by a word boundary on the right. This is one of several vowel reduction rules that apply to monosyllabic words in the BBN rule set. The symbol # for word boundary is a convention of the Bobrow-Fraser rule tester. The conventions for stress marking using the symbols !- (for reduced stress), !0 for zero stress, !1 for slightly more stress, and !2 for main stress are conventions adopted by the BBN speech group. The decision to make the stress levels uniformly increasing for increasing stress runs counter to the classical 1 for primary stress and 2 for secondary stress, but permits a uniform treatment of increasing levels of stress above a minimal level. The Bobrow-Fraser system permits the use of arbitrary such symbols to occur as elements of word pronunciations and in rules. The distinction between such symbols and phonemes is that special symbols have not been defined by the DP

operator to have a feature definition.

The Bobrow Fraser system provides an operator DT for defining a sequence of phonemes and special symbols (which they call a tree) on which to apply phonological rules. In our case, such trees will be base form pronunciations of words in our lexicon and will be defined by reading a dictionary file in a form which will be discussed later. However, for debugging purposes, it may be desirable to test a rule or rules on a tree which is typed in using the command as follows:

```
DT NEW (NIL # N UW !2 #)
```

which puts the list (NIL # N UW !2 #) under the property TREE on the property list of the word NEW. The format for such a tree entry consists of a sequence of phonemes and special symbols, bounded by word boundary symbols (#), and preceded by a list of features which can be used to enable the application of rules. In this case, the feature list is empty (NIL). (Caution, due to the implementation of these rules in the Bobrow-Fraser system, the last element of the tree cannot be changed or deleted, hence the final # is necessary.)

The Bobrow-Fraser system also provides for the definition of a tree as a list of other trees, a mechanism which can be used to apply a rule to each tree in the list.

However, this is not a convenient form for the expansion of a dictionary, and we have introduced a different mechanism for that task.

## 3. Extensions to the Bobrow-Fraser System

In order to use the Bobrow-Fraser system for systematically expanding a dictionary to be used as part of a speech understanding system rather than merely testing the operation of the rules on a few selected words, a number of additional features have been added. These included the ability to produce all possible application sequences where optional rules were involved, a variety of additional conditional rule application facilities, and several other features. In the remaining sections of this paper, we will focus on the details of these extensions and how they are being used in dictionary expansion and dictionary and rule debugging. We will describe where necessary some features of the Bobrow-Fraser rule notation, but for full details of the notation, see Bobrow and Fraser [1968].

## 4. Alternative Pronunciations

The Bobrow-Fraser system provides no mechanism for alternative pronunciations of the same word except as different named trees. Since in our speech dictionary we have words which require several alternative base-form

pronunciations to which phonological rules are to be applied, we have constructed our extensions to the Bobrow-Fraser system to handle an additional form of pronunciation tree consisting of an OR of several trees of the ordinary form as in the example:

```
(OR (NIL # N UW !2 #)(NIL # N Y UW !2 #))
```

5. Pronunciation Likelihoods

The purpose of expanding the dictionary for a speech understanding system by a set of phonological rules is to produce a set of all possible pronunciations of a word which the speech understanding sytem might be called upon to recognize. However, not all such pronunciations are equally acceptable, where by "acceptability" we refer to some measure of the goodness or correctness or likelihood of a particular pronunciation. To accommodate for this, we have adopted the convention that the first element of the features list which the Bobrow-Fraser system permits for each pronunciation will be used to hold a "pronunciation likelihood" that ranges from zero to one and is used as a measure of the quality of the pronunciation. Base-form pronunciations in the dictionary are given initial pronunciation likelihoods, which are then modified by multipliers associated with optional rules to reflect the relative acceptability of applying and not applying the

rule. Currently, the interpretation of these numbers is purely subjective, and we are attempting by looking at the behavior of questionable pronunciations in traces of overall system behavior to determine how these numbers should be assigned and used. A sample representation of a pronunciation tree with pronunciation likelihoods is:

(OR ((1.0) # N UW !2 #)((.8) # N Y UW !2 #))

## 6. Rule Trees

In the Bobrow-Fraser system, it is possible to define one rule as a set of other rules to be applied in sequence. However, the system does not provide a facility for specifying optional rules or for providing conditional application of rules based on the success or failure of previous ones. To handle these and other cases, we have implemented a set of routines which drive the rule tester from a structured organization of rules which we call a rule tree. In its simplest form, a rule tree is simply a list of rule names which are to be applied obligatorily in sequence. However, by embedding rules and rule trees in various rule tree operators (see below), one can obtain a powerful system for expressing conditional and optional application of rules. A sample rule tree containing a few rule tree operators is:

(R1 R2 (OPT R3)(IF R4 THEN R7 R8)(ELSE R6) R10)

This rule tree says to apply R1 followed by R2. Then if rule R3 matches the result of the first two applications (which may be the original base pronunciation if neither R1 nor R2 applied) both possibilities of applying the rule and not applying it should be considered. This produces two different pronunciations of the word. For each resulting pronunciation (only one if R3 did not match), R4 is attempted, and if it matches, then R7 and R8 are applied, otherwise not. If R4 fails to match, then R6 is attempted, otherwise not. Subsequently R10 is applied (whether or not R4 matched).

## 7. Rule Tree Operators

### (a) OPT

The rule tree operator OPT is used to indicate an optional rule which causes a phonological derivation to split into two cases when it matches. One case follows the derivation which makes the change indicated by the rule, and the other case proceeds as if the rule had not matched. Since the relative acceptability of the resulting pronunciations may not be the same, the general form or the OPT operator is (OPT rule n m) where n and m are numbers. To compute their respective pronunciation likelihoods, the likelihood of the input to the rule is multiplied by n for the case where the rule change is applied and by m for the

case where the rule matches but is not applied. When the rule fails to match, the pronunciation likelihood of the input is not changed. A typical specification of an optional rule would thus look like the following:

(OPT R3 1.0 .7)

indicating that the choice of not applying the rule is somewhat less acceptable than the choice of applying it. When no numbers are specified in an OPT statement, they are implicitly 1.0.

(b) IF

IF is used as an operator to indicate the conditional application of subsequent rules as a function of the success or failure of previous rules. When devising rules to capture phonological effects, it is possible to find a situation where one thing is to be done if a given rule applies, and something else if it does not. The general form of an IF statement is (IF rule THEN ruletree) The interpretation of an IF operator is that if the rule indicated applies, then the remaining rules inside the IF statement are tried before going on. If the rule does not match, then these rules are skipped. (In an earlier implementation, the interpretation was that these rules were the only rules to be tried if the rule succeeded and the rules outside the IF statement were the rules to be followed

if the rule failed. However, experience showed that the usual case of conditional rule use was that only the next rule or two was conditional on the test rule and that thereafter the two lists were the same. The current interpretation of the IF statement together with the ELSE statement thus provide a more compact and efficient representation of the usual case.)

(c) ELSE

The ELSE statement was introduced to immediately follow an IF statement and indicate those rules which are to be applied only if the test rule in the IF statement failed. Such a use of the ELSE statment was indicated in the introductory example. However, in many situations, the conditional behavior desired is that if a given rule fails, then some other rule is to be tried, but if it succeeds then no special rules are to be applied. This would result in an IF statement with empty consequences followe by an ELSE statement. To avoid the necessity to specify such an empty IF statment, the rule tree conventions permit the use of an ELSE command by itself to indicate a list of rules which are to be tried if the preceeding rule fails -- irrespective of whether the preceding rule is an IF statement. Such a use would be.

(R1 R2 (ELSE R3) R4 R5)

which says that after trying R1 and R2, R3 is to be tried if and only if R2 failed to match, and after that R4 and R5 are to be tried.

(d) MARK

The Bobrow-Fraser system permits, as we have said, a list of features to be associated with a pronunciation. Bobrow and Fraser called them syntactic features, but nothing in their implementation constrains them -- hence our ability to preempt the first one to use as a pronunciation likelihood. These features can be tested by the @ operator in the Bobrow-Fraser system in the context part of a rule specification to enable the application of a rule (i.e., their absence can be used to block a rule which requires them). If the first element of the context specification of a rule has the form (@ f1 ... fn), then that rule can only apply to a pronunciation which contains all of the features f1 ... fn. However, there is no provision in the Bobrow-Fraser system to add features to this list as a result of rule application. In our application, we would like to enable certain dialect- or even speaker-dependent rules to add marks to the pronunciation features list when they apply so that the knowledge of their application can be used by the higher level components of the speech understanding system. Hence the rule tree operator MARK was introduced to permit such additions to the pronunciation

features list. Its expected use is inside an IF statement to add the features if the test rule applies. A typical example is:

(R1 R2 (IF R3 THEN (MARK REDUCED)) R4 ... )

which adds the feature REDUCED if the rule R3 succeeds.

(e) TEST

Although the @ operator provides for the conditional rule application contingent on features associated with the base form pronunciation of a word (or added by the new MARK operator), one might want to make rule application conditional on other features of a word without having to represent these features redundantly on each possible base form pronunciation. The TEST operator allows one to make the application of a sequence of rules conditional on the success or failure of an arbitrary LISP predicate applied to the current word. The format of the TEST statement is identical to that of the IF statement except that instead of a test rule, the TEST statement has a LISP form which can have free references to the variable WORD (which is bound to the current word). An example would be:

(R1 R2 (TEST (GETP WORD (QUOTE AUX)) R3 R4) R5 ...)

which indicates that after R1 and R2 have applied, then R3 and R4 will be applied if and only if the form (GETP WORD (QUOTE AUX)) is true (i.e., the word is marked as having the

property AUX on its property list, which is used in our system to indicate membership in the syntactic category AUX). ELSE statements can be used following TEST statements just as they are following IF statements to indicate rules which are to be applied if and only if the preceeding test condition fails.

(f) STOP

Since the general convention for interpreting IF, ELSE, and TEST statements is that processing is to continue with the rules which follow the statement after the rules inside the statement have been done, the STOP operator is provided for the cases where this is not desired. When a STOP statement is encountered, subsequent application of rules ceases, and the current pronunciation at the time is taken as the final pronunciation. An example of the use of the STOP operator would be:

(R1 R2 (IF R3 THEN R4 R5 R6 (STOP)) R7 R8 R9)

which says that if R3 is successful, then R4, R5 and R6 are applied and no more, while if R3 fails, then R7, R8 and R9 are applied. An equivalent ruletree can be constructed using the ELSE operator instead of the STOP operator:

(R1 R2 (IF R3 THEN R4 R5 R6)(ELSE R7 R8 R9))

## 7. About Parentheses

In the above examples it no doubt became apparent that as the rule tree expressions become more complicated, the correct decipherment of the structure by counting and matching parentheses in a straight linear representation becomes tedious. This problem with list structures in LISP has long ago been solved by the development of programs which print out such structures in an indented form (called prettyprinting) which is pleasing to the eye and easy to read. The parenthesis notation, however, is a benefit for machine input, since it provides a format free means of entry (the user can use as much or as little format as he likes) while the prettyprint routines provide a formatted output automatically without the person having to laboriously count spaces (which is almost as tedious as counting parentheses). A prettyprinted representation of a rule tree is illustrated as follows:

```
(R1 R2 R3 (OPT R4 1.0 .7)
    (OPT R5)
    (OPT R6 .5 1.0)
    (IF (OPT R7 .5 .9)
        THEN R8 R9)
    (ELSE R10 (OPT R11))
    R12 R13 (TEST (GETP WORD (QUOTE N))
                  R14 R15)
    R16)
```

In the prettyprinted representation, successive elements of a list (a group of elements within a pair of parentheses) beyond the second element are aligned vertically under the second element, except that successive atomic elements (single rule names without parentheses) are printed on the same line until a non-atomic element (one with parentheses around it) is encountered. In other LISP systems, slightly different prettyprinting conventions are observed, and it is possible to write one's own prettyprinting routine for special applications if desired.

## 8. File Storage Conventions

Once a set of phonological rules and one or more rule trees has been defined, they can be saved on a file and loaded into the ruletester at a later time without redefining them. This is done by using the FILE function (a slightly modified version of the INTERLISP MAKEFILE function) and the prettydef commands PROPS:, which saves elements of property lists of atoms, and V:, which saves the value of an atom. The command

```
FILE(RULESET (PROPS: PHONEMES (PHONEME))
             (PROPS: RULES (RULE))
             (V: RULELISTS))
```

will construct a file named RULESET on which it will save the phoneme definitions (the property PHONEME on the property list of each atom in the list named PHONEMES), the

rule definitions (the property RULE on the property list of each element in the list named RULES), and each of the rule trees which is named in the list named RULELISTS. The lists RULES and PHONEMES are automatically maintained by the Bobrow-Fraser system as phonemes and rules are defined. The list RULELISTS must be set by the user using the LISP function SETQQ or SETQ just as he must use SETQQ or SETQ to define the individual r[illegible] trees. The first use of FILE will create an atom consisting of the file name plus the characters COMS which is bound to the list of prettydef commands used to save the file. Subsequent calls to save the file can simply give the file name with no commands and the previous commands list will be used. E.g. for the file name RULESET above, the command list RULESETCOMS will be created and saved on the file, so that subsequent to adding additional rules or editing them, a call FILE(RULESET) is sufficient to save everything indicated on RULESETCOMS.

The RULESETCOMS on the current BBN RULESET contains additional prettydef commands to initialize storage and to clear any PHONEMES or RULES properties left around from previous definitions, as well as to save comments associated with the rules.

## 9. Input Dictionary Format

The input for our dictionary expansions is taken from a file whose format reflects other uses in the BBN speech understanding system. This format consists of a set of property-value pairs for each word in the dictionary which are to be stored on the property list of the word. Each syntactic category for which the word is a member is entered on its property list with a value which specifies the orthographic inflections of the word as a member of that category (of interest only for nouns, verbs, and adjectives). In addition there are properties for various syntactic and semantic features and a PHONEMES property which gives the base form phonetic spelling (or spellings) of the word. These dictionary entries are printed on a dictionary file which the phonological rule component takes as input. An example of a dictionary entry for the word "charge" is:

```
[CHARGE
      (V S-D
       FEATURES (TRANS PASSIVE)
       N -S
       PHONEMES (CH AA !2 R JH))]
```

This indicates that the word is a verb which forms its written third person singular form by adding s and its past and past participle forms by adding d; that it has syntactic features marking it as transitive and passivizable; and that it is a noun which forms its plural by adding s.

## 10. Regular Inflections

In the current version of the dictionary expansion procedure, special provision has been made for automatically constructing and entering into the expanded dictionary the inflected forms of regularly inflected words. Specifically for a given word X, such inflected forms as are appropriate from the list X-ed, X-er, X-est, X-ing, and X-s are constructed and given phonetic spellings consisting of the phonetic spelling of the root word plus a special inflection symbol -ed, -er, -est, -ing, or -s. Phonological rules have been written which expand these special symbols into the appropriate phoneme sequences depending on the phonetic context in which they occur.

## 11. Editing

Since the definitions of rule trees are stored in the value cells of the LISP atoms which name them, the LISP function EDITV can be used to edit them. Details of the use of the LISP editing commands can be found in the INTERLISP manual [Teitelman, 1974]. The definitions of phonological rules are stored on the property list of the rule name under the property name RULE. These can be changed using the LISP function EDITP.*

One can also alter the feature definitions of a phoneme using EDITP to edit the PHONEME property of the phoneme, but

because the order of the phonemes seems to be important, the resulting system may not work quite properly unless the RULESET file is saved and reloaded.

## 12. Using the Rule Expander

The normal mode of using the rule expander is to start with a dictionary file of the form used in the BBN speech system, including a PHONEMES property which gives the phonetic spelling or spellings of a word. If sufficiently small, this dictionary can be loaded into core by a function GETDICT, and subsequent phonological rule expansion performed in core. The current system can accept a dictionary up to about 600 words in this form, with the resulting expanded dictionary consuming about 47K of core.

For larger dictionaries, dictionary expansion can be done incrementally from an external file with the results of

- - - - - - - - - -

*The Bobrow-Fraser system makes use of a property FRULE, which contains a compiled form of the rule for use by the FLIP system which matches the rules. This property should be removed whenever a rule is edited, which can be done automatically with a piece of advice on the EDITP function. A new FRULE property is added automatically whenever a rule match is attempted for a rule which does not have one, thus getting an FRULE property which is consistent with whatever change has been made to the rule during the editing.

the expansion being written to an output file and not retained in core. For debugging purposes, a dictionary expansion produces summary results of the application of rules in the form of a property USEDRULES for each word which lists the rules which were applied to that word, and a list RULEHISTORY which lists for each rule those words to which the rule applied. The expansion programs also compute the total number of resulting pronunciations and the expansion ratio which results from the application of optional rules. Two output files are generally produced, an EXPDICT file which contains all of the syntactic properties and the expanded pronunciations and is used as input to the lexical retrieval component of the speech system, and an EXPHONES file which contains the original and expanded pronunciations and the RULEHISTORY and USEDRULES information for visual inspection by the phonologist developing the rules.

Also for debugging purposes, a function EXPWORD is provided which will apply a rule tree of phonological rules to a single word (with the possibility of setting a trace flag to watch the individual applications of rules) in order to test out immediately the consequences of an intended rule change or to track down the details of what happened when the effect of a rule tree on a word is not what was anticipated.

## 13. Implementation

The modifications which we have made to the Bobrow-Fraser system have been confined to a set of user-callable functions which interface to the Bobrow-Fraser routines via a single function TRYRULE (which calls their function DORULES with appropriate arguments and context). The Bobrow-Fraser system is in turn based on a pattern-matching system called FLIP [Teitelman, 1967] which is loaded into LISP. The upshot of these several levels of indirection is that the system is not particularly fast. The application of a single rule to a pronunciation takes on the order of .3 cpu seconds, but with 56 rules, the expansion of a single word takes approximately 17 seconds on the average, and the expansion of a large dictionary can take hours of CPU time. For interactive debugging, the amount of time to trace individual words is not excessive, and for full expansions, the system is run overnight when our machine is otherwise lightly loaded. Moreover, full expansions are not a frequent necessity, though if the system were to be used extensively, a new implementation not based on FLIP would be appropriate.

## 14. Rule Expansion Functions

The various functions which are called to use the rule tester are:

(a) GETDICT (dictfile)

GETDICT is a function which reads a dictionary from an external file and constructs the TREE properties on which the rule expander operates from the PHONEMES property of each word. The format for the PHONEMES property in the dictionary takes one of three forms: 1) a single list of phonemes for a single spelling, 2) a list whose first element is a pronunciation features list for a single spelling with pronunciation features (the pronunciation features list may or may not start with a pronunciation likelihood), or 3) a list of the form ((OR p1 p2 ... pn)), where each pi is a pronunciation of form 1 or 2. The resulting TREE property will always have a pronunciation features list preceding each pronunciation and a pronunciation likelihood as the first element in this list (1.0 is used if none was specified in the PHONEMES property).

GETDICT also constructs the inflected forms of regularly inflected nouns, verbs, and adjectives as described above and places on their property list the property ROOT-INFLECT which lists the root word of which this word is an inflection, the inflectional ending added, and the syntactic categories of which this word is a member. GETDICT sets the value of the variable WORDS to the list of words which result (both read from a file and constructed by regular inflection) and associates a numerical WORDINDEX property with each word that is used is a numerical reference to the word by the speech understanding system.

(b) EXPLIST (wordlist ruletree traceflag)

EXPLIST takes a list of words to be expanded as its first argument, a ruletree to use for the expansion as its second argument, and a trace flag which indicates whether or not to type out a trace of the intermediate rule matching attempts. It results in the addition of two new properties to each word in wordlist: EXPHONES which gives the expanded pronunciations and USEDRULES which lists the rules used. It also sets the variable RULEHISTORY to a record of the words to which each rule applied, and it sets the variables #roots, #words, and #pronunciations to the number of input words, the number of words resulting after addition of regularly inflected forms, and the total number of pronunciations of all words, respectively. The results are left in core where they can either be inspected on line, or saved by means of the FILE command. The command FILE(EXPDICT (PROPS: WORDS EXPROPS)) can be used to save the expanded dictionary, while the command FILE(EXPHONES (PROPS: WORDS (PHONEMES EXPHONES USEDRULES)) (V: (RULEHISTORY #ROOTS #WORDS #PRONUNCIATIONS))) can be used to save the summary results for inspection.

(c) EXPDICT (dictfile ruletree traceflag)

EXPDICT is a function which packages together a call to GETDICT to load the dictionary followed by a call to EXPLIST to expand it, followed by commands to save an EXPDICT file and an EXPHONES file. It leaves the results in core in addition to making these files, so that the appropriate context is already defined for using EXPWORD to examine the behavior of the rules on individual words (i.e., the TREE property is still on each word as are all its syntactic features).

(d) EXPWORD (word ruletree traceflag)

EXPWORD is a function for use in debugging rules. It expands the word given as its first argument using the rule tree indicated by its second argument. EXPWORD is generally called with the traceflag T (whereas the other EXP functions are usually called with traceflag NIL) in order to see a trace of the individual rule attempts. This trace shows the initial pronunciation of the word, each attempt to match a rule, the results of all successful rule matches, and the alternative rule matching paths that result from the success of optional rules. An example of such a trace is given in Section 15.

(e) EXPFILE (filename ruletree traceflag)

EXPFILE is similar to EXPDICT except that it operates incrementally from an external dictionary file instead of loading the file into core and operating on it there. Otherwise its results are the same as EXPDICT except that nothing is left in core as a result of the expansion. EXPFILE has been successfully used on a vocabulary of 1000 root words and should have no significant upper limit on vocabulary size.

(f) GETWORD (word file)

Since EXPFILE does not leave an environment in core from which to run EXPWORD for debugging, GETWORD has been provided for use in this situation to bring in from an external file the properties of a word that are necessary for expansion (i.e., the TREE property plus any others that might be tested by a predicate in a TEST statement). If the second argument specifying the input file is not specified, GETWORD will use the filename stored in the variable EXTEMPFILE which is set by EXPFILE when it operates to remember the file on which it stored its temporary results (This is the normal way to use GETWORD in a core image which was made as a SYSOUT after a call to EXPFILE).

## 15. An Example

To illustrate the operation of the rule expansion component, we give here an example of a call to EXPWORD with the traceflag set and a sample rule tree DLIST = (R46 R16 (OPT Z16 1.0 1.0) (OPT Z56 1.0 .2)(OPT R50 1.0 .5)(OPT R30.Z .5 1.0) R55):

```
62_EXPWORD(EVERY DLIST T]
(RULEHISTORY reset)
((1.0) # EH !2 * V AXR !- * IY !0 #)
R46
R16
Z16
((1.0) # EH !2 * V AX !- * R IY !0 #)
Z56
R50
((1.0) # EH !2 * V * R IY !0 #)
R30.Z
R55
((1.0) # EH !2 V * R IY !0 #)
((1.0) # EH !2 V * R IY !0 #)

FROM:
((.5) # EH !2 * V AX !- * R IY !0 #)
R30.Z
R55
((.5) # EH !2 * V AX !- * R IY !0 #)

FROM:
((1.0) # EH !2 * V AXR !- * IY !0 #)
Z56
R50
R30.Z
R55
((1.0) # EH !2 * V AXR !- * IY !0 #)

PRONUNCIATIONS:
  (((1.0)
    # EH !2 V * R IY !0 #)
   ((.5)
    # EH !2 * V AX !- * R IY !0 #)
   ((1.0)
    # EH !2 * V AXR !- * IY !0 #))
EVERY
```

Each trace path begins with a base-form or intermediate pronunciation and ends with a final pronunciation; there are as many trace paths as there are final pronunciations of the word (due either to alternative base forms or optional rules). The name of each rule is printed out as it is considered, and if it succeeds, then the result of the application is printed out immediately afterwards. (If the last rule in the list is successful, this results in two adjacent typeouts of the same pronunciation -- one for the successful rule application and the one that normally terminates a trace path.) Here, the rules Z16 and R50 are optional while rule R55 is obligatory, with R50 succeeding only on the path or which Z16 was successful. This resulted in three final pronunciations which are summarized at the end of the trace. Notice that each time an optional rule is taken, a later trace path will begin from the point where that rule applied, pursuing the alternative of not applying the rule. The symbol * is used in these rules as a syllable boundary marker.

There is a quirk of the trace which does not show here that when an optional rule is traced, the pronunciation likelihood for the immediate result of that application is not updated correctly, but will show correctly in the next rule application or in the final pronunciation. This is due to the inability to get inside the Bobrow-Fraser implementation to change the number before the trace is

printed out. In the current example, both optional rules have 1.0 likelihood multipliers when they apply so the effect is not apparent.

## 16. Conclusions

We have described a set of additions and extensions to the Bobrow-Fraser rule tester which make it into a useful tool for debugging and testing systems of rules on a large dictionary and for expanding such a dictionary for practical use as part of a speech understanding system. Although the rule application is relatively slow, a dictionary expansion needs only to be done occassionally and does not, of course, affect the operating speed of the speech understanding system that uses its output. The major features of our extended rule tester are its ability to systematically produce the alternative pronunciations which result from optional rules, its abilty to take specifications of complex rule systems using conditional operators to determine what rules to try when, and its features for keeping records of the history of rule applications during large dictionary expansions for use in evaluating and debugg ng rule systems.

The extended rule tester has been used for dictionary expansions for about six months at the present writing, with the features currently available gradually evolving over that period. Both for rule testing and debugging and for dictionary expansion it has proven to be a useful facility.

## References

Bobrow, D.G. and J.B. Fraser (1968)
"A Phonological Rule Tester, CACM, Vol. 11, No. 11, November.

Chomsky, N. and M. Halle (1968)
"The Sound Patterns of English," Harper and Row, New York.

Teitelman, W. (1967)
"Design and Implementation of FLIP, a LISP Format Directed List Processor," BBN Science Report No. 10, Bolt Beranek and Newman Inc., Cambridge, Ma.

Teitelman, W. (1974)
"INTERLISP Reference Manual," Xerox Palo Alto Research Center, Palo Alto, Calif.

## D. Audio Response Generation

Craig Cook
Bertram Bruce
Laura Gould

### 1. Introduction

During the past quarter, an audio response generation component was added to SPEECHLIS by connecting text response generation programs in the Retrieval component to our synthesis-by-rule program. Hereafter referred to as "TALKER", the audio response generation component combines the flexibility of natural English response with that of spoken output.

Response generation is the final critical link in a long process that begins with the speaker making an utterance. But the response is more than just the end product; it affects the speaker's way of describing entities in the domain and can tell him much about the capabilities of the system and how it operates.

### 2. Generation of English Responses from the Semantic Network

#### a. Overview

TALKER can be invoked in two principal ways. In one, the function CTALKER takes a typed-in English sentence which it converts into SPEECHLIS dictionary format using the

function SPEECHIFY. It then produces a phonetic transcription of the sentence using the most likely pronunciation for each of the words and calls the speech synthesis program to generate a waveform file corresponding to the phonemes. In the other mode, the function OUTPUT: takes a concept in the semantic network, a name referring to a set of appropriate syntactic templates, and a flag indicating desired length of output, and instantiates one of the templates to the concept. OUTPUT: then linearizes the resulting parse tree, applying a few special purpose transformations (e.g., tense changes on the verb) and inserting prosodic cues for the speech synthesis program. (These cues derive from the structure of the syntactic template, which is one reason for not working solely with linearized trees.) From there the action is similar to that of CTALKER. SPEECHIFY is called, the phonemes are obtained, and the waveform is generated. This section discusses the functions currently used in producing English responses and presents some examples from the travel budget semantic network.

b. SPEECHIFY

The function SPEECHIFY is an essential part of both modes of response generation, and is also being used now in the text version of the Travel Budget Manager's Assistant. SPEECHIFY performs three major types of conversions. The

first is to break up inflected words. Using the algorithm in Winograd [1971], SPEECHIFY converts "budgeted" to "budget-ed", "trips" to "trip-s", "Wolf's" to "Wolf s" (where "s" is a dictionary entry for the possessive morpheme), and so on. In each case the converted forms contain only words in the dictionary. The second type of conversion is to replace numbers (including cardinals, ordinals, digit strings, and monetary figures) with strings of number words. For example, "$354.78" becomes "three hundred and fifty four dollar-s and seventy eight cent-s", "3547 miles" becomes "three thousand five hundred and forty seven mile-s", "trip number 3547" becomes "trip number thirty five forty seven", "October 8th, 1975" becomes "October eighth nineteen-m seventy five" (where "nineteen-m" is the dictionary entry for the modifier form of "nineteen," see Section I.D.), and "account 11273" becomes "account one one two seven three". Note that these conversions require context sensitive checks on the surrounding words since, for example, amounts of money are referenced in a different way from trip numbers. The third conversion performed by SPEECHIFY involves a number of special words represented idiosyncratically in the SPEECHLIS dictionary. For example, multi-word compounds need to be concatenated, e.g., "Los Angeles" becomes "Los@Angeles", and homographs need to be distinguished, e.g., "estimated" becomes "estimate-v-ed" (where "estimate-v" is the dictionary entry for the verb

"estimate" and "estimate-n", the entry for the noun form).

c. GETPHONEMES

Following action by SPEECHIFY, the function GETPHONEMES obtains phonemes for each word in the sentence. If a parse tree exists for the sentence, GETPHONEMES can also add stress and pause markers. For example, it inserts the symbol "FW" to indicate reduced stress in a following function word, and the symbol ")N", to indicate a major phrase boundary. GETPHONEMES currently extracts the pronunciation of highest probability from the expanded phoneme dictionary. Again, this procedure is only approximately correct, and we hope to devise procedures which will be more sensitive to the sentence and discourse context.

d. OUTPUT:

The top level program for response generation is OUTPUT:. OUTPUT: produces the sentence which is used by SPEECHIFY and GETPHONEMES to build a phoneme string for speech synthesis. The input to OUTPUT: is a semantic network node, a constituent type and a flag indicating desired length of the response. The node should be an INSTANCE/OF a general type (e.g., a particular trip is an INSTANCE/OF the concept of DB/TRIP). Associated with the type node is a set of language generation trees representing

the base forms of syntactic constituents for describing the instance nodes. Depending upon the length of response flag, the constituent desired, and other possible conditions (see Example 1 below), OUTPUT: selects a language generation tree and fills it in with information from the semantic network. Once the tree is filled, transformations may be applied, for example, to change the verb form from present to past.

e. Language Generation Trees

A language generation tree may be a completely specified parse with every leaf being an English word. In that case, OUTPUT: merely linearizes the tree so that it may be printed out or used to synthesize speech. On the other hand, a leaf in a tree may be a semantic network link name, indicating that the node at the end of the link is to be described as the stated constituent. ("End of the link" is used here to mean the value found after application of both direct lookup and default inference procedures.) For example, a tree branch, (NP TRAVELER), means "find the TRAVELER for this node and construct a noun phrase referring to him". Also, a leaf may be "!", indicating that the node is to be described first as its own sub-constituent. For example, the tree for describing a time period as a sentence, contains the branch, (PP !), meaning "describe the time period as a prepositional phrase". This latter feature often simplifies the writing of the trees. A final leaf

type in the trees is the special form, (FUNCTION <function name> <link>), indicating that the named function is to be applied to the node at the end of link. This allows things to be stored in one fashion and described in another.

In the remainder of this section, examples of nodes in the semantic network, trees for data base types, generated English text, and phoneme strings are shown to demonstrate current uses of these response generation programs. We hope both to extend the generation language in the coming year and to remove current ad hoc features such as the handling of verbs.

f. Example 1

The first item shown below is the type node for "conference", followed by a particular instance of a conference, the 13th ACL Meeting:

```
637 - DB/CONFERENCE
   CREATOR          CHIP
   TYPE             LINKNAME
   LINKS            (TIME) (LOCATION) (SPONSOR)
                    (CREATE/TIME)
                    (DB/CREATOR) (ATTENDED/BY)
   INSTANCES        462 579 888 961 965 972 976 979
                    1029 1034 1041
   METHODS          [Follow back pointers from time point
                     to db/conference 956]
   VALUE/CLASS/OF   (TO/ATTEND) (REGISTRATION/FEE/OF)
   ENGLISH/NAME     (CONFERENCE)

462
   CREATOR          CHIP
   INSTANCE/OF      (DB/CONFERENCE)
   SPONSOR          (ACL)
   LOCATION         (BOSTON)
```

```
   TIME                546
   DB/CREATOR          [CHIP BRUCE 276]
   REGISTRATION/FEE    901
```

Next are the time nodes [Woods et al., 1975c, pp. 15-26] and the registration fee associated with this conference:

```
546
   CREATOR             CHIP
   INSTANCE/OF         (TIME/PERIOD)
   BEGIN/TIME          461
   END/TIME            536
   TIME/OF             462
   DURATION            991

461
   CREATOR             CHIP
   HOUR                9
   YEAR                1975
   MONTH#              10
   DAY/OF/MONTH        30
   INSTANCE/OF         (TIME/POINT)
   BEGIN/TIME/OF       546
   PRECEDES            536
   PRECEDED/BY         1105

536
   CREATOR             CHIP
   HOUR                12
   YEAR                1975
   MONTH#              11
   DAY/OF/MONTH        1
   INSTANCE/OF         (TIME/POINT)
   END/TIME/OF         546
   PRECEDES            885
   PRECEDED/BY         461

901
   CREATOR             CHIP
   VALUE               20
   INSTANCE/OF         (DB/FEE)
   DB/CREATOR          [CHIP BRUCE 276]
   REGISTRATION/FEE/OF 462
   CREATE/TIME         957
```

Using these nodes and the one-element list of trees for "registration fee" -

```
((S (NP (DET "The")
        (ADJ "registration")
        (N "fee")
        (PP (PREP "for")
            (NP REGISTRATION/FEE/OF)))
    (VP (V "is")
        (NP (ADJ (FUNCTION GPRIN1 VALUE))
            (N UNITS]
```

OUTPUT: produces the following English text string (note that UNITS was obtained via a default inference procedure) -

> The registration fee for the ACL conference is 20 DOLLARS.

SPEECHIFY produces the list of dictionary entries -

> (THE REGISTRATION FEE FOR THE ACL CONFERENCE IS TWENTY DOLLAR-S)

and GETPHONEMES produces the list of phonemes -

```
(FW # DH AH !2 # R EH !1 * JH IX !- * S T R EY !2 * SH EN !-
# F IY !2 # FW # F AO !2 R # DH AH !2 # EY !1 * S IY !1 * EH
!2 L # K AA !2 N * F AXR !- * EN !- S # IH !2 Z # T W EH  !2
N * T IY !0 # D AA !2 * L AXR !- Z %.)
```

Next is a pair of trees for "conference", one for a description of a conference as a sentence and one for a description of a conference as a noun phrase. Note the use of the "!" convention in the first tree. Also note the use

of the *OPT* flag to indicate that the (PP TIME) constituent can be omitted if the TIME of the conference is not found in the semantic network.

```
((S (NP !)
    (VP (V "will be held")
        (PP (PREP "in")
            (NP LOCATION))
        (PP TIME *OPT*)))
 (NP (DET "the")
     (ADJ SPONSOR)
     (N "conference")))
```

Using the above nodes for the ACL conference and these trees, OUTPUT: produces -

> The ACL conference will be held in BOSTON from October 30th to November 1st, 1975.

SPEECHIFY produces the list of dictionary entries -

> (THE ACL CONFERENCE WILL BE HELD IN BOSTON FROM
> OCTOBER THIRTIETH TO NOVEMBER FIRST NINETEEN-M
> SEVENTY FIVE)

and GETPHONEMES produces the phoneme list (note that the "FW" is omitted before the first function word since the subsequent word starts with a vowel) -

```
(DH AH !2 # EY !1 * S IY !1 * EH !2 L # K AA !2 N * F AXR !-
*  EN !- S # W IH !2 L # B IY !2 # HH EH !2 L D # FW # IH !2
N # B AO !2 * S T EN !- # F R AH !2 M # AX !- K * T OW !2  *
B AXR !- # TH ER !2 * DX IY !0 * IX !- TH # FW # T UW !2 # N
OW !0 * V EH !2 M * B AXR !- # F ER !2 S T # , # N AY !2 N *
T IY !1 N # S EH !2 * V AX !- N * T IY !0 # F AY !2 V %.)
```

Next are trees for "time period". Here a new feature of the "trees" language appears. Associated with each tree may be a condition which must be satisfied before the tree can be used. If the tree is a list beginning with the constituent name (as are all the trees shown above), then the condition is null (and therefore vacuously true). Otherwise the tree is a list of two lists; one, the tree itself, and the other, the condition on applicability of the tree. OUTPUT: finds the first tree whose condition holds and which can be completely filled out. To fill out a tree, every link and ! must determine a fillable subtree. (The fill requirement can be relaxed by use of the *OPT* flag mentioned above.) In this case there are four trees for descriptions of time periods as prepositional phrases and one for a description of a time period as a sentence. The variables A and B are used as registers in the conditions. NPY is an invented constituent used for time points with the year specified, e.g., "on March 5th, 1975."

```
((S (NP (DET "the")
        (N "time")
        (PP (PREP "of")
            (NP TIME/OF)))
    (VP (V "is")
        (PP !)))
```

```
((PP (PREP "from")
     (NP BEGIN/TIME)
     (PREP "to")
     (NPY END/TIME))
 (AND (SETQ A (GET: ITEM (QUOTE BEGIN/TIME)
                    NIL
                    (QUOTE DONTASK)))
      (SETQ B (GET: ITEM (QUOTE END/TIME)
                    NIL
                    (QUOTE DONTASK)))
      (NEQ A B)))
((PP (PREP "on")
     (NPY BEGIN/TIME))
 A)
((PP (PREP "on")
     (NPY END/TIME))
 B)
((PP (PREP "for")
     (NP DURATION))
 (AND (GET: ITEM (QUOTE DURATION)
            NIL
            (QUOTE DONTASK]
```

OUTPUT: produces the sentence -

> The time of the ACL conference is from October 30th to November 1st , 1975.

SPEECHIFY produces the list of dictionary entries -

> (THE TIME OF THE ACL CONFERENCE IS FROM OCTOBER THIRTIETH TO NOVEMBER FIRST , NINETEEN-M SEVENTY FIVE)

and GETPHONEMES produces the list of phonemes -

```
(FW # DH AH !2 # T AY !2 M # AH !2 V # FW # DH AH !2 # EY !1
* S IY !1 * EH !2 L # K AA !2 N * F AXR !- * EN !- S # IH !2
Z # F R AH !2 M # AX !- K * T OW !2 * B AXR !- # TH ER !2  *
T Y IX !- TH # FW # T UW !2 # N OW !0 * V EH !2 M * B AXR !-
# F ER !2 S T # , # N AY !2 N * T IY !1 N # S EH !2 *  V  AX
!- N * T IY !0 # F AY !2 V %.)
```

g. Example 2

Next are semantic network nodes for "fare" with three instances of fares. This is followed by the tree for "fare"

and examples of responses. Note the use of the *OPT* flag in the tree to indicate that the mode of transportation is optional.

```
485 - DB/FARE
   LINKS              (TYPE) (MODE/OF/TRANSPORT)
                      (CREATE/TIME) (DB/CREATOR)
                      (VALUE) (FARE/OF)
   INSTANCES          159 477 798 949 950 951 952 953
                      954 955 1054 1069 1070 1072 1073
                      1075 1077 1078 1079 1081 1083 1085
                      1087 1091 1093 1095 1096 1097 1098
                      1099 1100 1135
   ENGLISH/NAME       (FARE)

951
   CREATOR            CHIP
   VALUE              40.8
   INSTANCE/OF        (DB/FARE)
   DB/CREATOR         [CHIP BRUCE 276]
   MODE/OF/TRANSPORT  (BUS)
   FARE/OF            555
   CREATE/TIME        713

555
   CREATOR            CHIP
   MILEAGE            496
   MEMBERS            (BOSTON) (PITTSBURGH)
   INSTANCE/OF        (CITY/PAIR)
   FARE               951 952 1095

952
   CREATOR            CHIP
   VALUE              57
   INSTANCE/OF        (DB/FARE)
   DB/CREATOR         [CHIP BRUCE 276]
   MODE/OF/TRANSPORT  (AIR)
   FARE/OF            555
   CREATE/TIME        713

1095
   CREATOR            LAURA
   VALUE              41.5
   INSTANCE/OF        (DB/FARE)
   DB/CREATOR         [LAURA GOULD 501]
   CREATE/TIME        957
   MODE/OF/TRANSPORT  (TRAIN)
   FARE/OF            555
```

```
[(S
   (NP
     (DET "The")
     (ADJ (NP (ADJ "one")
              (N "way")))
     (ADJ MODE/OF/TRANSPORT *OPT*)
     (N "fare"))
     (PP (PREP "from")
         (NP [N (FUNCTION (LAMBDA (X)
                    (DNAME (GET: X (QUOTE STARTING/POINT)
                     NIL (QUOTE DONTASK]
     (PP (PREP "to")
         (NP (N (FUNCTION (LAMBDA (X)
                    (DNAME (GET: X (QUOTE DESTINATION)
                     NIL (QUOTE DONTASK]
   (VP (V "is")
       (NP (ADJ (FUNCTION GPRIN1 VALUE))
           (N UNITS]
```

Below are the outputs of OUTPUT:, SPEECHIFY, and GETPHONEMES for each of the three fares to Pittsburgh.

The one way BOSTON to PITTSBURGH BUS fare is 40 DOLLARS.

```
(FW # DH AH !2 # W AH !2 N # W EY !2 # B AO !2 * S T EN !- #
FW # T UW !2 # P IH !2 T S * B ER !1 G # B AH !2 S # F EH !2
R # IH !2 Z # F AO !2 R * DX IY !0 # D AA !2 * L  AXR  !-  Z
%.)
```

The one way BOSTON to PITTSBURGH AIR fare is 57 DOLLARS.

```
(FW # DH AH !2 # W AH !2 N # W EY !2 # B AO !2 * S T EN !- #
FW # T UW !2 # P IH !2 T S * B ER !1 G # EH !2 R # F EH !2 R
# IH !2 Z # F IH !2 F * T IY !0 # S EH !2 * V EN !- #  D  AA
!2 * L AXR !- Z %.)
```

The one way BOSTON to PITTSBURGH TRAIN fare is 41 DOLLARS.

```
(FW # DH AH !2 # W AH !2 N # W EY !2 # B AO !2 * S T EN !- #
FW  # T UW !2 # P IH !2 T S * B ER !1 G # T R EY !2 N # F EH
!2 R # IH !2 Z # F AO !2 R * DX IY !0 # W AH !2 N # D AA  !2
* L AXR !- Z %.)
```

## 3. Speech Synthesis

Given the phonetic spelling of a string of words together with syntactic markers (non-phonetic symbols), a mapping program translates them from the phoneme set of the phonetic dictionary into that of the synthesis program. The syntactic markers, such as the "FW" and ")N" mentioned above, are used to make changes to the input string at the phonetic level.

Inside the synthesis-by-rule program, the phonological phase processes the input string, removing the non-phonetic input symbols. Rule application requires that the input string be processed both left-to-right and right-to-left. This phase contains all of the program's phonetic rewrite rules plus a number of acoustic-phonetic rules dealing with phonological effects. The input string minus the non-phonetic symbols next enters the Phonetic phase which processes it left-to-right. This phase contains approximately 100 acoustic-phonetic rules which accomplish the phonetic-to-parametric conversion into a set of acoustic parameter tracks. The waveform synthesizer routine then generates a waveform from these parameter tracks and outputs it to a disk file. The speech generation process is now complete. The synthesis-by-rule program is entirely implemented in software on our PDP-10. Although this is computationally expensive (ca. 35 times real-time), we feel

it is reasonable for our current purposes. In addition, we have the option of using special-purpose hardware at a later date should we desire a near real-time implementation.

4. Discussion

Since TALKER became operational, we have generated over ten minutes of synthetic speech. This effort has benefited other components of SPEECHLIS in the following ways:

1) We have uncovered several errors in the phonetic dictionary having to do with spellings and pronunciation likelihoods. Missing words have also been discovered (i.e., "way" as in "one way fare").

2) Audio response has provided an impetus to exercise all aspects of the retrieval and inference routines which drive the audio generation. This has provided an important additional check on data base consistency and correctness.

3) We have done extensive testing of the synthesis-by-rule program which is critical to the operation of the verification component. By using the travel budget dictionary, we have tested those words and phonetic contexts which will appear as spoken input to SPEECHLIS. Problems of pronunciation have appeared and resulted in changes to the synthesis program. Certain acoustic-phonetic and phonological rules have also required modification.

Although we have not dealt extensively with prosodic issues, we have made an effort to improve the naturalness of the synthetic speech. In TALKER's input set there are a number of non-phonetic symbols, including lexical stress marks, commas, periods, and question marks plus syllable, word and phrase boundary markers. Since a complete syntactic structure of the generated response is available, we simply insert the punctuation and syntactic markers into the input string at the appropriate places. Informal listening tests have provided a measure by which we can then judge the overall naturalness and intelligibility.

## References

Woods, W.A., R. Schwartz, C. Cook, D. Klatt. J. Wolf, L. Bates B. Nash-Webber, B. Bruce, and J. Makhoul (1975b) "Speech Understanding Systems, Quarterly Technical Progress Report No. 2, 1 February 1975 to 1 May 1975," Report No. 3080, Bolt Beranek and Newman Inc., Cambridge, Ma.

Woods, W.A., R. Schwartz, C. Cook, J. Klovstad, L. Bates, B. Nash-Webber, B. Bruce, J. Makhoul (1975c) "Speech Understanding Systems, Quarterly Technical Progress Report No. 3, 1 May 1975 to 1 August 1975," Report No. 3115, Bolt Beranek and Newman Inc., Cambridge, Ma.

Winograd, T. (1971) "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", Project MAC Report, TR-84, MIT.